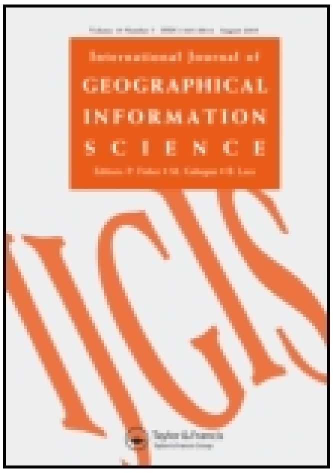

International Journal of Geographical Information Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tgis20

Generalized network Voronoi diagrams: Concepts, computational methods, and applications

A. Okabe [a] , T. Satoh [b] , T. Furuta [c] , A. Suzuki [d] & K. Okano [e]

[a] Center for Spatial Information Science , University of Tokyo , c/o Department of Urban Engineering , 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

[b] Application Technology Development, Research and Development Center , Pasco Corporation , 2-8-10 Higashiyama, Meguro-Ku, Tokyo, 153-0043, Japan

[c] Department of Management Science , Tokyo University of Science , 1-14-6 Kudan Kita, Chiyoda-Ku, Tokyo , 102-0073, Japan

[d] Department of Information Systems and Mathematical Sciences , Nanzan University , 27 Seirei, Seto, 489-0863, Japan

[e] 5-7-20 Tsurumaki, Tama-shi, Tokyo 206-0034, Japan

Published online: 15 Jul 2008.

International Journal of Geographical Information Science
Vol. 22, No. 9, September 2008, 965–994

Research Article

Generalized network Voronoi diagrams: Concepts, computational methods, and applications

A. OKABE*†, T. SATOH‡, T. FURUTA§, A. SUZUKI¶ and K. OKANO‖

†Center for Spatial Information Science, University of Tokyo, c/o Department of Urban Engineering, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
‡Application Technology Development, Research and Development Center, Pasco Corporation, 2-8-10 Higashiyama, Meguro-Ku, Tokyo, 153-0043, Japan
§Department of Management Science, Tokyo University of Science, 1-14-6 Kudan Kita, Chiyoda-Ku, Tokyo 102-0073, Japan
¶Department of Information Systems and Mathematical Sciences, Nanzan University, 27 Seirei, Seto, 489-0863, Japan
‖5-7-20 Tsurumaki, Tama-shi, Tokyo 206-0034, Japan

(*Received 11 November 2006; in final form 23 July 2007*)

In the real world, there are many phenomena that occur on a network or alongside a network; for example, traffic accidents on highways and retail stores along streets in an urbanized area. In the literature, these phenomena are analysed under the assumption that distance is measured with Euclidean distance on a plane. This paper first examines this assumption and shows an empirical finding that Euclidean distance is significantly different from the shortest path distance in an urbanized area if the distance is less than 500 m. This implies that service areas in urbanized areas cannot be well represented by Voronoi diagrams defined on a plane with Euclidean distance, termed *generalized planar Voronoi diagrams*. To overcome this limitation, second, this paper formulates six types of Voronoi diagrams defined on a network, termed *generalized network Voronoi diagrams*, whose generators are given by points, sets of points, lines and polygons embedded in a network, and whose distances are given by inward/outward distances, and additively/multiplicatively weighted shortest path distances. Third, in comparison with the generalized planar Voronoi diagrams, the paper empirically shows that the generalized network Voronoi diagrams can more precisely represent the service areas in urbanized areas than the corresponding planar Voronoi diagrams. Fourth, because the computational methods for constructing the generalized planar Voronoi diagrams in the literature cannot be applied to constructing the generalized network Voronoi diagrams, the paper provides newly developed efficient algorithms using the 'extended' shortest path trees. Last, the paper develops user-friendly tools (that are included in SANET, a toolbox for spatial analysis on a network) for executing these computational methods in a GIS environment.

Keywords: Voronoi diagram; Network; Shortest path distance; Directed distance; Weighted distance; *k*th nearest point; Farthest point

*Corresponding author. Email: atsu@ua.t.u-tokyo.ac.jp

1. Introduction

The Voronoi diagram (abbreviated to VD) has been extensively studied in various fields and generalized in many different ways since the pioneering work by Voronoi (1908) (a review is provided by Okabe *et al.* (2000)). Originally, the VD was defined for a set of points $P=\{p_1,\ldots, p_n\}$ (termed *generator points* or *generators*) on a plane, where the distance $d(p, p_i)$ between an arbitrary point p and a generator point p_i on the plane is the Euclidean distance. In these terms, the (*ordinary*) *planar VD* (P-VD) is defined as a set of polygons, $\mathrm{Vor}=\{\mathrm{Vor}_1,\ldots, \mathrm{Vor}_n\}$, where the polygon Vor_i is given by

$$\mathrm{Vor}_i=\{p|d(p,p_i)\leq d(p,p_j), j\neq i, j=1,\ldots,n\} \tag{1}$$

An example P-VD is illustrated in figure 1, which will be compared with VDs defined on a network in section 2.

The objective of this paper is to generalize the P-VD by replacing the plane with network space (such as a road network) and the Euclidean distance with distances defined on a network (such as the shortest-path distance). Such VDs are termed *generalized network Voronoi diagrams* (N-VDs).

Generalized N-VDs have many potential applications in various fields, including marketing, operations research, traffic-accident analysis, criminology, and animal and plant ecology. For example, Miller (1994), Okabe and Kitamura (1996), Morita *et al.* (2001), and Okabe and Okunuki (2001) dealt with market area delimitation within networks using GIS; Furuta *et al.* (2006) proposed an optimization method for ambulance stations on a road network in Seto; Yamada and Thill (2004) analysed traffic accidents in Buffalo (see also Jones *et al.* 1996, Levine *et al.* 1995, McGuigan 1981, Nicholson 1989); Spooner *et al.* (2004) investigated roadside acacia populations on a road network; O'Driscoll (1998) investigated the distribution of seabirds along a coastline; Bashore *et al.* (1985), Mallick *et al.* (1998), Clevenger *et al.* (2003), and Saeki and MacDonald (2004) examined animal kills on roads; and Painter (1994), Bowers and Hirschfield (1999), Ratcliffe and McCullagh (1999), Anselin *et al.* (2000), and Ratcliffe (2002), analysed street crimes.

Generalized N-VDs can also be utilized in methods for spatial analysis. For example, Okabe *et al.* (1995) developed the network nearest-neighbour distance

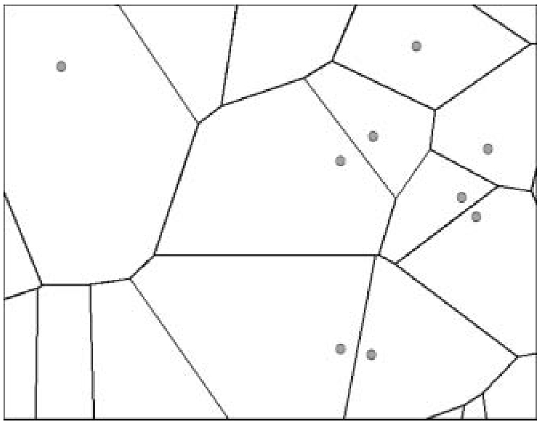

Figure 1. Part of the ordinary P-VD generated by parking lots in Kyoto (the whole study area is shown in figure 5).

method; Okabe and Yamada (2001) developed the K function method for a network; Satoh and Okabe (2006a) formulated the network nearest-neighbour distance method for lines and polygons using the line/polygon N-VD; and Satoh and Okabe (2006b) proposed the local network Voronoi K-function method by extending the local planar Voronoi K-function method (Okabe *et al.* 2006a).

As is noted from these examples, generalized N-VDs are potentially and actually useful, but the P-VD with Euclidean distance is frequently utilized as a first approximation to service areas of facilities, although actual service is achieved through a network. Some reasons for this frequent use might be:

1. it is believed that the shortest-path distance can be approximated by the corresponding Euclidean distance;
2. managing spatial data is more difficult on a network than on a plane.

The second reason remains true, but Geographical Information Systems (GIS) greatly relieve this difficulty. In addition, network data as well as detailed spatial data are now widely available. However, the first reason looks doubtful in urbanized areas. Maki and Okabe (2005) demonstrated that shortest-path distances are significantly different from Euclidean distances in Kokuryo, a suburb of Tokyo, if the Euclidean distance is less than 500 m (figure 2). Table 1 shows that the average radii of many types of stores in the Shinjuku ward, a subcentral ward in Tokyo, are less than 500 m. Therefore, to estimate service areas of facilities in urbanized areas, a VD defined on a network is more appropriate than the P-VD.

The VD on a network is defined for a set of points $P=\{p_1,\ldots, p_n\}$ on the network, where the distance $d\,(p, p_i)$ between an arbitrary point p and a generator point p_i on the network is measured using the shortest-path distance, $d_{\mathrm{S}}\,(p, p_i)$. In these terms, the (*ordinary*) *network* VD (N-VD) is defined as a set of subnetworks, (referred to as *Voronoi subnetworks*), $\mathrm{Vor}=\{\mathrm{Vor}_1,\ldots, \mathrm{Vor}_n\}$, where the Voronoi subnetwork Vor_i is given by

$$\mathrm{Vor}_i=\{p|d_{\mathrm{S}}(p,p_i)\le d_{\mathrm{S}}(p,p_j),\, j\ne i, j=1,\ldots,n\} \tag{2}$$

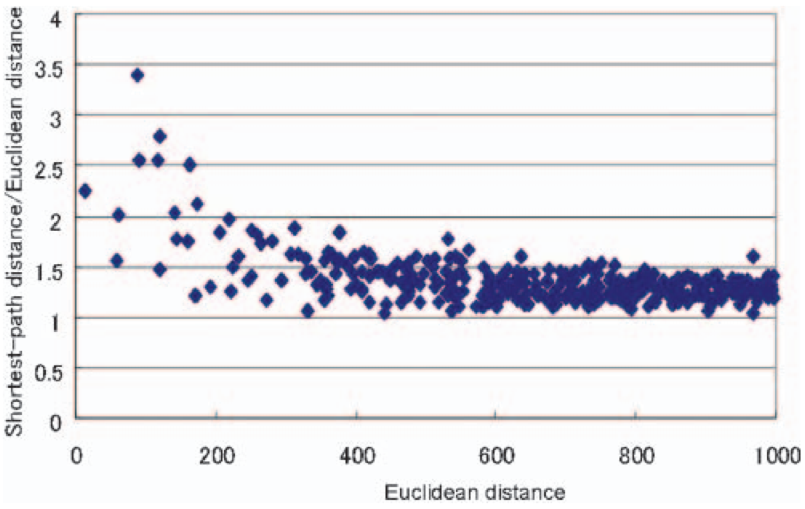

Figure 2. Ratio of the shortest-path distance to Euclidean distance on the street network in Kokuryo, a suburb of Tokyo (from Maki and Okabe 2005).

Table 1. Average service radii of stores in Shinjuku ward, Tokyo.

Store types	Average radius (m)
Bakery	320
Shoe store	255
Fruit shop	213
Book store	177
Chinese noodle shop	153
Convenience store	150
Beauty parlour	114

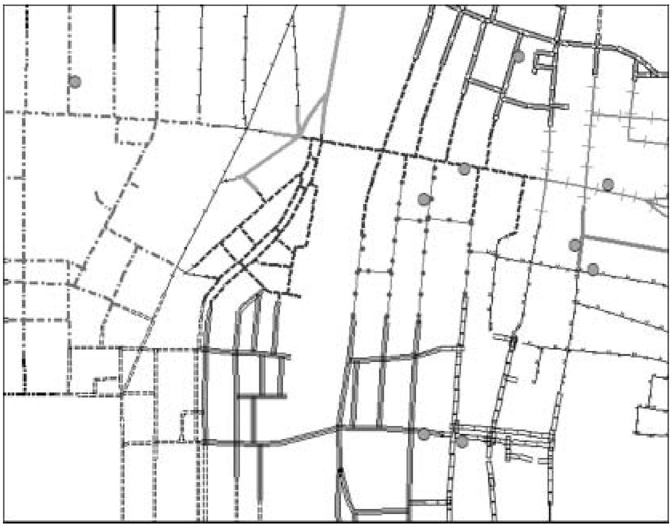

Figure 3. Part of the ordinary N-VD generated by parking lots in Kyoto, which correspond to the ordinary N-VD shown in figure 2 (the whole study area is shown in figure 5).

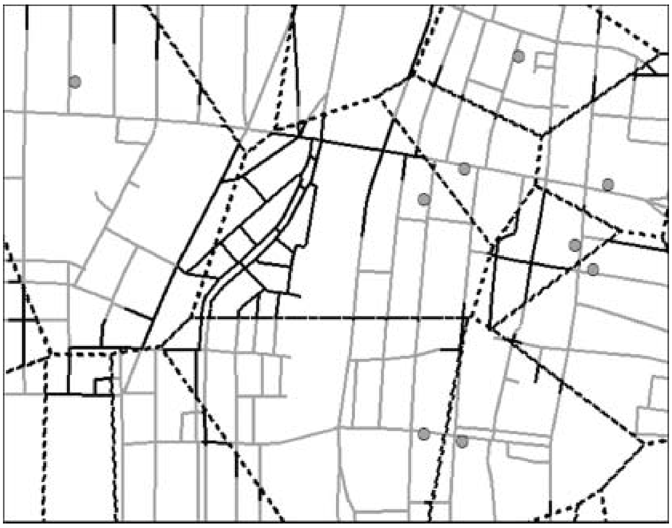

Figure 4. Streets (the black lines) of the Voronoi subnetworks of the N-VD (figure 3) that are not included in corresponding Voronoi polygons of the P-VD (figure 1) (the generators are the same).

An actual example of the N-VD is shown in figure 3, where the generators are parking lots in Kyoto, which corresponds to the P-VD in figure 1. To clearly show the difference between figures 2 and 3, figure 4 is depicted, where the black lines indicate the streets of the Voronoi subnetworks of the N-VD (figure 3) that are not included in corresponding Voronoi polygons of the P-VD (figure 2) (the generators are the same).

As shown in Okabe *et al.* (2000), the ordinary P-VD has been generalized in many different ways with respect to space (including Cartesian, Riemann, sphere, cylinder, cone, torus, and polyhedral surface), distance (including Euclidean distance, Manhattan distance, Karlsruhe distance, Hausdorff distance, supremum metric, elliptic distance, and convex distance), and generators (such as points, lines, polygons, sets, and moving points). In contrast, the ordinary N-VD has been little generalized; in fact, no generalizations of the N-VD were found in Okabe *et al.* (2000). Because the approximation of the shortest-path distance by Euclidean distance is inappropriate in urbanized areas (recall figure 2), generalized N-VDs cannot be substituted by the corresponding generalized P-VDs. Moreover, microscopic spatial analysis is increasingly demanded because in urbanized areas, stores with small market areas (shown in table 1) become very competitive and precise spatial analysis is expected. Motivated by these facts, we develop generalized N-VDs in this paper.

This paper consists of five sections. Section 2, formulates six types of N-VD. These N-VDs are fairly similar to P-VDs in concept, but the computational methods for constructing these N-VDs are completely different from those for P-VDs. Therefore, section 3 develops their computational methods. As mentioned above, we expect these N-VDs to be used not only by professional spatial analysts but also by non-professional analysts, such as managers of retail stores, who are not always good at developing computer programs. To support non-professional spatial analysts, section 4 implements the computational methods developed in section 3 as user-friendly GIS-based tools. The paper concludes in section 5, with a summary of the results.

2. Formulation of generalized Voronoi diagrams

2.1 *Directed network Voronoi diagrams*

The ordinary N-VD defined in section 1 assumes that the network is a *non-directed network*, i.e. the network consists of two-way streets. In reality, especially in urbanized areas, street networks are *directed networks,* i.e. they include one-way streets. An actual example in Kyoto is shown in figure 5, where one-way streets are indicated by black lines, and two-way streets are indicated by grey lines. Many streets in urbanized areas (especially in downtown areas) are one-way. This regulation greatly affects the service areas of facilities. For instance, delivery bikes of pizza stores choose the shortest path from their stores to their consumers considering one-way regulations. Therefore, to estimate market areas in urbanized areas, N-VDs formulated on directed networks are indispensable.

There are two distinctive differences between directed N-VDs and non-directed N-VDs. The first difference is that distances on a non-directed network are symmetrical, i.e. $d_S(p, p_i)=d_S(p_i, p)$ always holds, while distances on a directed network, $d_{\vec{S}}(p,p_i)$, are asymmetrical, i.e. $d_{\vec{S}}(p,p_i)=d_{\vec{S}}(p_i,p)$ does not always hold. On a directed network, referring to p_i, the distance $d_{\vec{S}}(p_i,p)$ is referred to as the *outward*

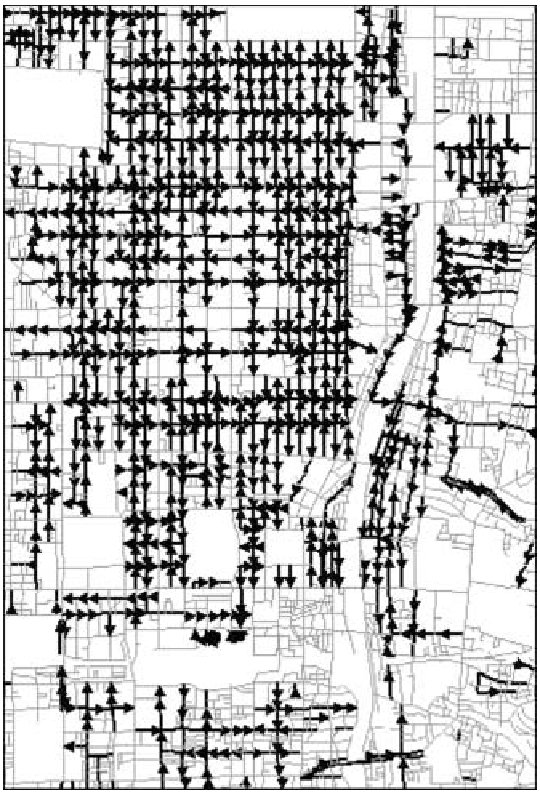

Figure 5. Street network system in Kyoto (the black lines indicate one-way streets, and the grey lines indicate two-way streets).

distance from p_i, and the distance $d_{\vec{S}}(p,p_i)$ is referred to as the *inward distance to* p_i. In terms of the outward and inward distances, directed N-VDs are formulated in two ways: a directed N-VD, Vor$=\{\mathrm{Vor}_1,\ldots,\mathrm{Vor}_n\}$, defined with the inward distance, i.e.

$$\mathrm{Vor}_i=\{p|d_{\vec{S}}(p,p_i)\leq d_{\vec{S}}(p,p_j),\, j\neq i, j=1,\ldots,n\} \tag{3}$$

and a directed N-VD defined with the outward distance, i.e.

$$\mathrm{Vor}_i=\{p|d_{\vec{S}}(p_i,p)\leq d_{\vec{S}}(p_j,p),\, j\neq i, j=1,\ldots,n\} \tag{4}$$

(notice that $d_{\vec{S}}(p,p_i)$ is used in equation (3), but $d_{\vec{S}}(p_i,p)$ in equation (4)). The former N-VD is termed the *inward N-VD*, and the latter N-VD is termed the *outward N-VD*. The outward N-VD is useful, for example, for examining the coverage area of emergency facilities, such as fire stations, because the shortest path from a fire station p_i to a burning house at p is crucial. The inward N-VD is useful, for example, for estimating the service areas of parking lots, because the shortest path from a driver at p to the nearest parking lot p_i is crucial. Figures 6 and 7 show an outward N-VD and an inward N-VD, respectively. Note that the outward/inward N-VDs are not naturally defined for P-VDs but are inherent in the N-VDs.

Figures 8 and 9 show the difference between the outward/inward N-VDs and their corresponding P-VDs. The lines in these figures indicate the lines of each Voronoi subnetwork that are not included in the corresponding Voronoi polygon of the

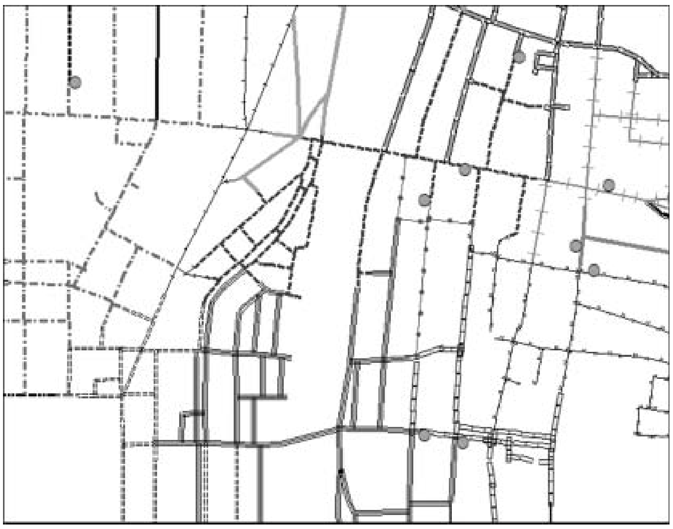

Figure 6. Part of the outward N-VD generated by parking lots in Kyoto (the whole study area is shown in figure 5).

P-VD. As these figures indicate, the difference between directed N-VDs and their corresponding P-VDs is large. Comparison between figures 8 and 9 reveals that the difference between the outward N-VD and the inward N-VD is also large. In addition, figure 10 shows the difference between the inward N-VD and the non-directed N-VD for all of Kyoto. The difference indicated by the bold lines in figure 10 amounts to 29% of the total length of the street network.

A second difference between the non-directed N-VD and the directed N-VD is that the former N-VD always forms a tessellation (i.e. $\mathrm{Vor}_1,\ldots, \mathrm{Vor}_n$ are mutually exclusive except at the boundary points and collectively exhaustive), but the latter

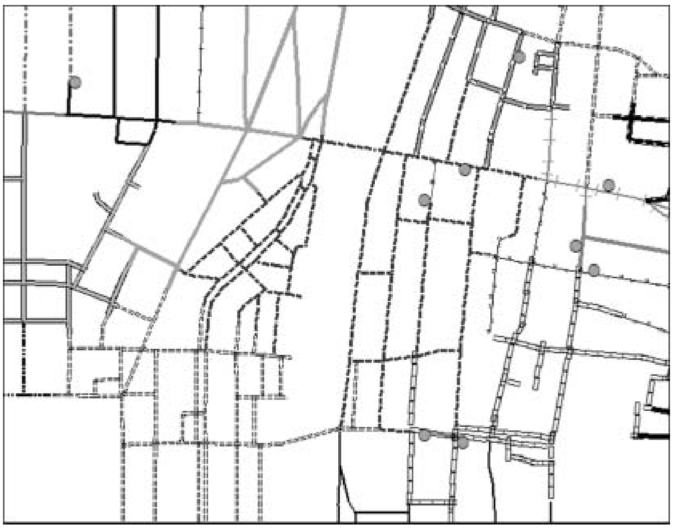

Figure 7. Part of the inward N-VD generated by parking lots in Kyoto (the whole study area is shown in figure 9).

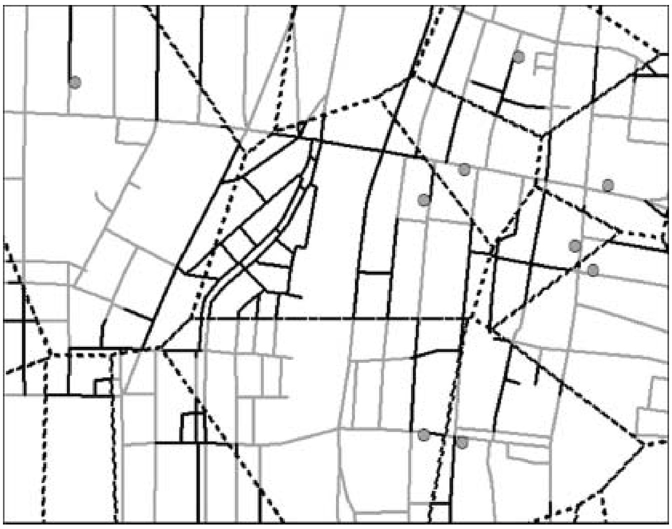

Figure 8. Line segments of each Voronoi subnetwork of the outward N-VD that are not included in the corresponding Voronoi polygon of the P-VD (figure 1).

may not form a tessellation. On a specific directed network, there may be links that are not accessible from points on other links (imagine streets not accessible to cars). Note that this difference does not affect their computational methods (section 3).

2.2 *Weighted network Voronoi diagrams*

The family of weighted VDs is often adopted in market area analysis. According to Shieh (1985), one of the earliest uses of weighted P-VDs dates back to Rau (1841). A rigorous application of the weighted P-VD to market area analysis was developed by Boots (1980), and there have been many applications since then. However, compared

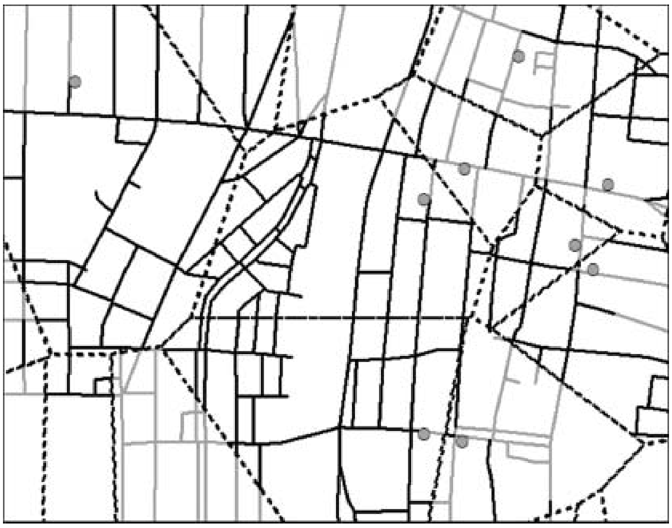

Figure 9. Line segments of each Voronoi subnetwork of the inward N-VD that are not included in the corresponding Voronoi polygon of the P-VD (figure 1).

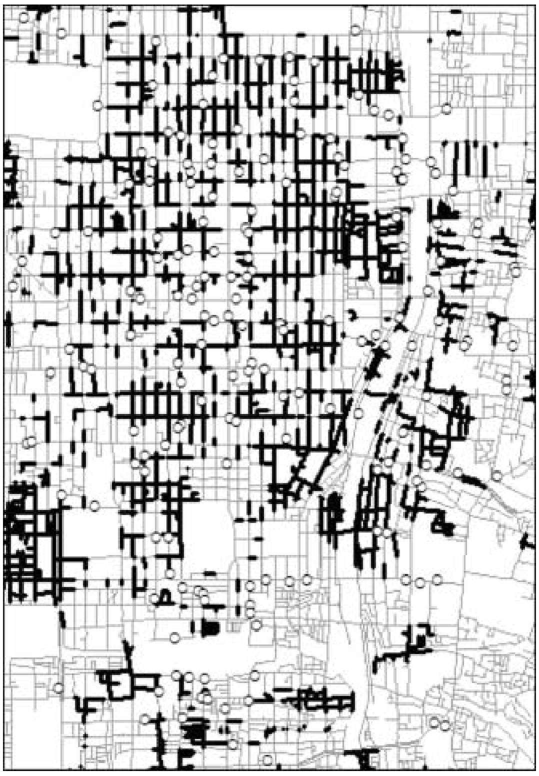

Figure 10. Difference between the inward N-VD and the nondirected N-VD (the bold lines).

with the weighted P-VD, the weighted N-VD is rarely found in the related literature. Conceptually, the extension of the weighted P-VD to the weighted N-VD is straightforward, but some extensions are inherent in a network.

To give a practical example, consider consumers who want to choose a store to buy goods. They consider not only the distance between their houses and the nearest stores, but also the prices of goods at the stores. Therefore, the 'distance' is measured by

$$d_{\mathrm{AW}}(p,p_i) = d_{\mathrm{S}}(p,p_i) + \beta_i \tag{5}$$

where $d_{\mathrm{S}}(p, p_i)$ is the shortest-path distance or transportation cost, and β_i is a constant for the generator point p_i, representing the price level at p_i. This distance is called the *additively weighted distance* (Okabe *et al.* 2000). In this term, let Vor_i be the set of points (a Voronoi subnetwork) on a network defined by equation (2) where $d_{\mathrm{S}}(p, p_i)$ is replaced with $d_{\mathrm{AW}}(p, p_i)$ of equation (5). The resulting set of Voronoi subnetworks, $\mathrm{Vor} = \{\mathrm{Vor}_1, \ldots, \mathrm{Vor}_n\}$ is termed the *additively weighted N-VD*. An example is depicted in figure 11.

When the unit cost α_i of delivering goods from store p_i varies, the *multiplicatively weighed distance* (Okabe *et al.* 2000),

$$d_{\mathrm{MW}}(p,p_i) = \alpha_i\, d_{\mathrm{S}}(p,p_i) \tag{6}$$

is appropriate. In this term, the *multiplicatively weighed N-VD* is formulated as equation (2), where $d_{\mathrm{S}}(p, p_i)$ is replaced with $d_{\mathrm{MW}}(p, p_i)$ of equation (6). An example is shown in figure 12.

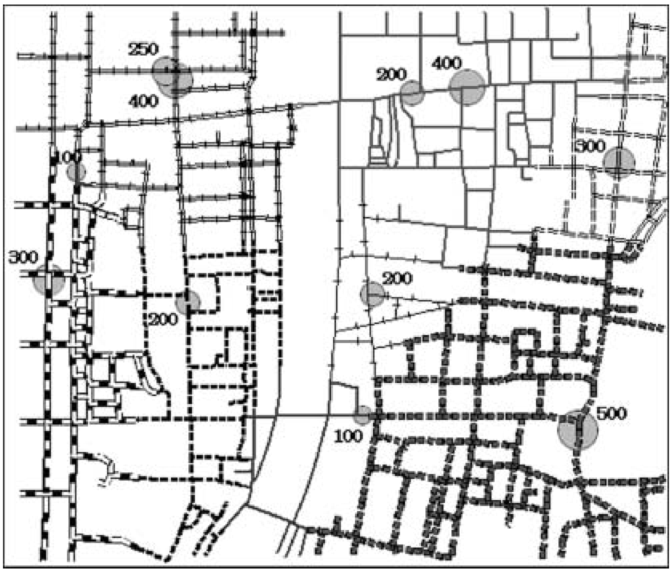

Figure 11. Part of the additively weighted N-VD generated by convenience stores in Kyoto (the weights are represented by the radii of circles, and the whole study area is shown in figure 5).

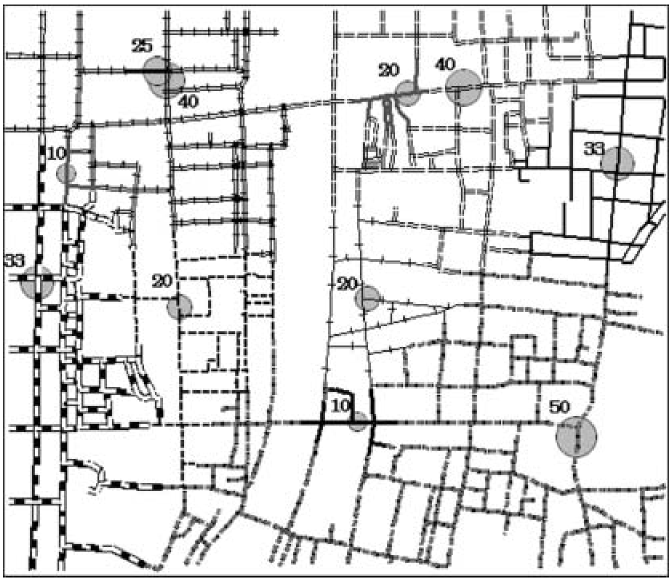

Figure 12. Part of the multiplicatively weighed N-VD generated by convenience stores in Kyoto (the weights are represented by the radii of circles, and the whole study area is shown in figure 5).

The weighted N-VDs on a non-directed network share the same concept with the weighted P-VDs, but the weighted N-VDs on a directed network are inherent in a network. As shown in section 2.1, *inward*- and *outward-weighted N-VDs* can be defined on a directed network. Another interesting property that is enjoyed by the

N-VDs, but not by the P-VD, is that the computational method of the multiplicatively weighed N-VD easily produces the *farthest-point multiplicatively weighed N-VD* with a minor modification. This property will be discussed in section 3.

2.3 *kth nearest point N-VD and farthest point N-VD*

The ordinary N-VD considers the nearest generator points, and so it can alternatively be called the *nearest point N-VD*. This N-VD can be extended to the k*th nearest point N-VD*. To be explicit, let $\Pi(P\backslash\{p_i\})$ be the power set of $P\backslash\{p_i\}$ (the complement of $\{p_i\}$ with respect to P), and S be a subset of $\Pi(P\backslash\{p_i\})$ satisfying that the number of elements in S is $k-1(k\geqslant 2)$. Let Vor_i be

$$\mathrm{Vor}_i = \bigcup_{\{S\in\Pi(P\backslash\{p_i\}),|S|=k-1\}} \{p|d_{\mathrm{S}}(p_h,p)\leq d_{\mathrm{S}}(p_i,p),p_h\in S \text{ and } d_{\mathrm{S}}(p_i,p)\leq d_{\mathrm{S}}(p_j,p),\, p_j\in P\backslash(S\cup\{p_i\})\} \quad (7)$$

Then, the set $\mathrm{Vor}=\{\mathrm{Vor}_1,\ldots, \mathrm{Vor}_n\}$ is called the kth nearest point N-VP (Furuta *et al.* 2005). The kth nearest point N-VD is used when (k – 1)th nearest facilities are out of service or exceed their capacities. When $k=n$, it is called the *farthest point* N-VD. Note that when $k=n$, the second condition in equation (7) becomes $P\backslash(S\cup\{p_i\})=\phi$, implying that the second condition is dropped. Almeida and Güting (2006) and Kolahdouzan and Shahabi (2005) used the term kth nearest-neighbours, but their problem was to find the kth nearest-neighbours when a point is moving, and this problem is completely different from the kth nearest point N-VD discussed here.

Compared with the farthest-point P-VD, the farthest-point N-VD is especially useful for evaluating max/min-type facilities accessed by cars, such as ambulance stations and fire stations, because the shortest path to a patient's house or a burning house on a street network is crucial. For max/min facilities, the *outward farthest-point*

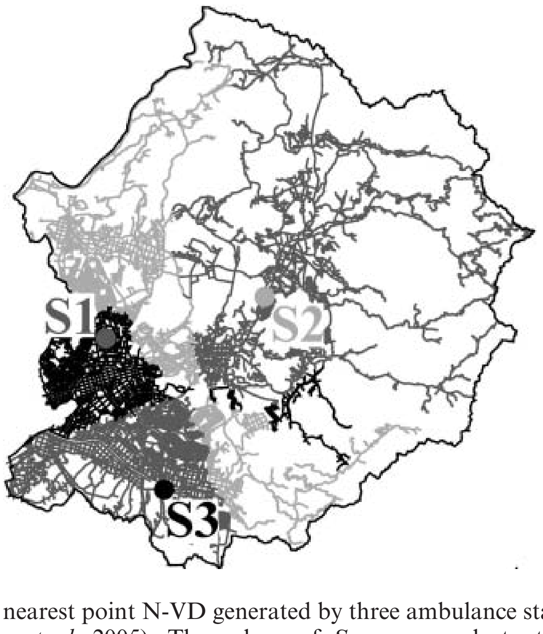

Figure 13. Second nearest point N-VD generated by three ambulance stations (S_1, S_2, S_3) in Seto, Japan (Furuta *et al.* 2005). The colour of S_i corresponds to that of its Voronoi subnetwork Vor_t.

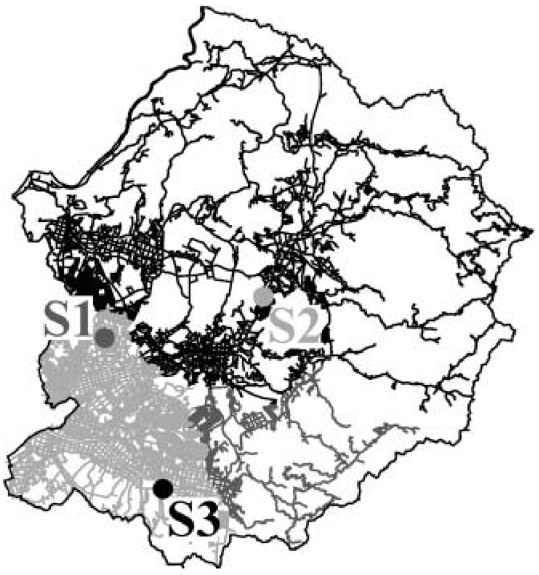

Figure 14. Farthest-point N-VD generated by three ambulance stations (S_1, S_2, S_3) in Seto, Japan (Furuta *et al.* 2005) (the colour of S_i corresponds to that of its Voronoi subnetwork Vor_i).

N-VD defined on a directed network should be used. Examples of the second-nearest-point N-VD and the farthest-point N-VD for ambulance stations in Seto, Japan, are shown in figures 13 and 14, respectively. In the same way, the *inward farthest-point N-VD* can be defined.

2.4 *Line N-VD*

To analyse the effect of line-like facilities, for instance, the effect of arterial streets on the distribution of street crimes, the N-VD generated by the set of subnetworks $L=(L_1,\ldots,L_n)$ (generators) is useful, because street crimes occur on streets, and criminals flee along streets.

Let $p_{i1},\ldots,p_{im_i}$ be the nodes of a subnetwork L_i (generators) and define the distance $d_S(p, L_i)$ between a point p and a generator subnetwork L_i as

$$d_s(p, L_i) = \min\{d_S(p,p_{i1}),\ldots,d_S(p,p_{im_i})\} \tag{8}$$

Let Vor_i be the set of points on a network defined by equation (2) where $d_S(p, p_i)$ is replaced with $d_S(p, L_i)$ of equation (8). Then, the set $\mathrm{Vor}=\{\mathrm{Vor}_1,\ldots, \mathrm{Vor}_n\}$ is termed the *line N-VD*. For a directed network, the *inward and outward line N-VD* can be defined with $d_{\vec{S}}(p,L_i)$ and $d_{\vec{S}}(L_i,p)$, respectively. Examples are shown in figure 15, where the coloured lines in figure 15(*a*) indicate the arterial streets in Kyoto (the grey lines indicate non-arterial streets); figures 15(*b*) and 15(*c*) show the outward and inward line N-VD generated by the arterial streets, respectively (the colours in (*a*) correspond to those in (*b*) and (*c*)).

2.5 *Polygon N-VD*

To analyse the effect of polygon-like facilities, for instance, the effect of parks on the distribution of high-class apartment buildings, the N-VD generated by the set of

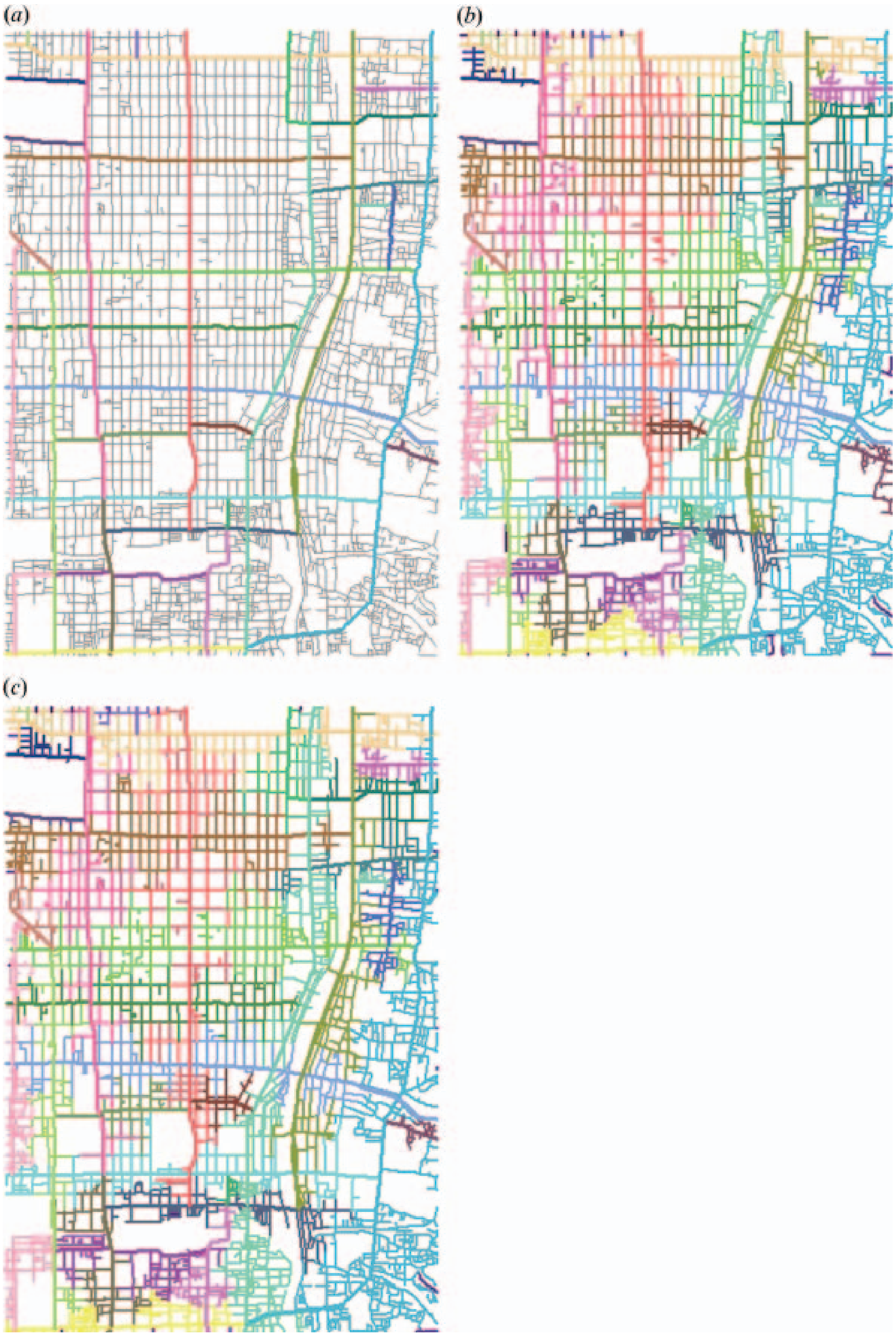

Figure 15. Arrial streets (the coloured lines) (*a*); outward (*b*) and inward (*c*) line N-VDs generated by the arterial streets in Kyoto (the colours in (*a*) correspond to those in (*b*) and (*c*)).

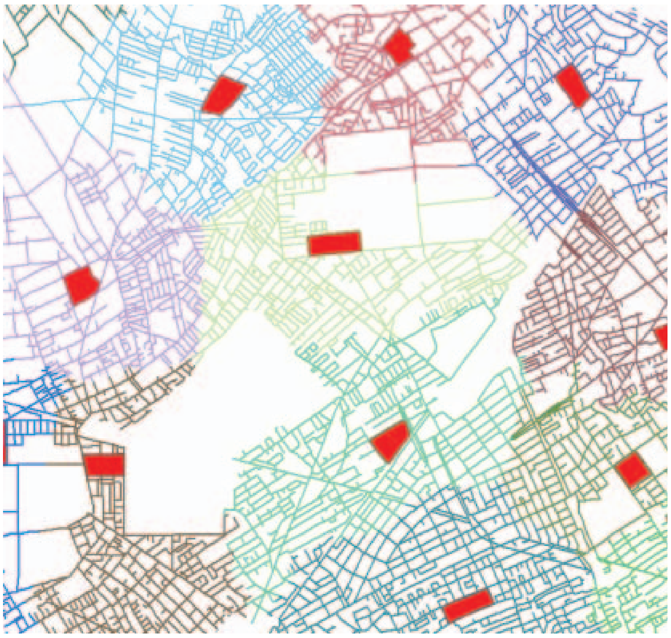

Figure 16. Polygon N-VD generated by elementary schools (the red polygons) in Sagamihara, Japan.

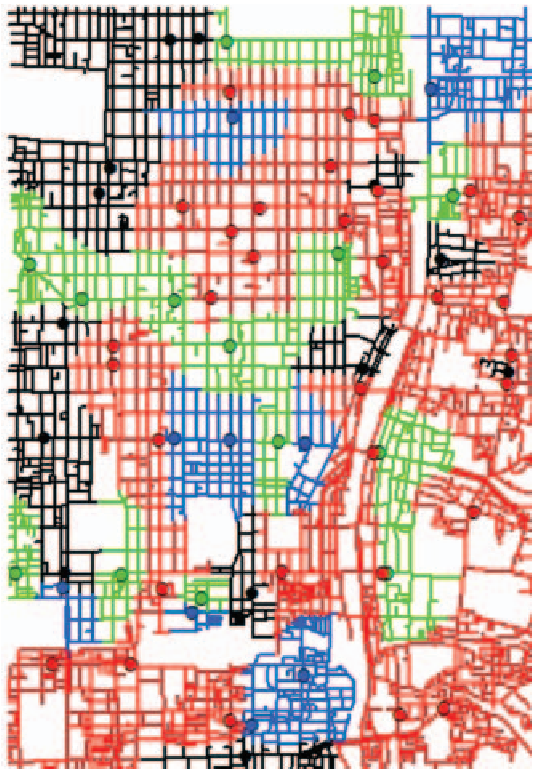

Figure 17. Point-set N-VD generated by chain stores (7-Eleven (black circles), Lawson (blue circles), CircleK (green circles), and others (red circles)) in Kyoto.

polygons $A=\{A_1,\ldots, A_n\}$, called the *polygon N-VD*, is useful. On a network, because a polygon is represented by a loop surrounding the polygon, and the loop is a special case of a generator subnetwork L_i, the mathematical definition is the same as the line N-VD. An example is illustrated in figure 16, where the polygons are elementary schools in Sagamihara, Japan.

2.6 *Point-set N-VD*

The last generalization is simple but useful in practice. Chain stores in cities, Starbucks and Tully's for instance, are competing for their market areas. To estimate their market areas, let $P_i=\{p_{i1},\ldots,p_{im_i}\}$ be a set of points (stores) belonging to the *i*th class (*i*th chain store; a generator), and $P=\{P_i,\ldots, P_n\}$. The distance d_S (p, P_i) between p and P_i is measured by

$$d_S(p,P_i)= \min\{d_S(p,p_{ik})|p_{ik}\in P_i\} \tag{9}$$

Let Vor_i be the set of points on a network (a Voronoi subnetwork) defined by equation (2) where d_S (p, p_i) is replaced with d_S (p, P_i) of equation (9). Then, the N-VD generated by P is termed the *point-set N-VD*. An example point-set N-VD is shown in figure 17, where sets of points (generators) represent the chains of convenience stores (7-Eleven, Lawson, CircleK, and others).

The point-set N-VD is also useful for microscopic spatial analysis for large stores. Large stores, such as department stores, have more than one entrance. Large parks are also often accessible through several entrances. In these cases, access points to a facility are represented by a set of points, $P_i=\{p_{i1},\ldots,p_{im_i}\}$ (a generator), and its service area is approximated by the Voronoi subnetwork of P_i of the point-set N-VD.

3. Computational methods

This section develops computational methods for constructing the N-VDs formulated in the preceding section.

3.1 *Extended shortest-path tree*

Computational methods for constructing N-VDs differ depending on their exact type, but they share a common technique, the *shortest-path tree* (SPT), defined as the set of edges connecting all vertices such that the sum of the edge lengths from a given root vertex to each vertex is minimized. An example is shown in figure 18, where a network, directions and the SPT rooted at v_i, SPT(v_i), are indicated by the continuous line segments, the arrows along line segments, and the heavy line segments, respectively.

As can be seen in figure 18, the SPT should be modified to construct N-VDs, because the SPT spans all vertices, but does not cover all edges. The edges indicated by the hairline segments in figure 18, which are referred to as *uncovered edges*, are not included in the SPT. To cover them, two types of auxiliary vertices are generated on uncovered edges. The first type is generated on uncovered two-way edges. On each edge, there exists a point (termed a *break vertex*; the small white circles in figure 19) from which there are two different shortest paths to the root vertex of the SPT (e.g. $v_2\rightarrow v_4\rightarrow v_5\rightarrow b_2$ and $v_2\rightarrow v_{11}\rightarrow v_6\rightarrow b_2$). At the break vertices, each edge is divided at its break vertex; as a result, two edges are obtained (e.g. e_2 in figure 18 is divided at b_2 into e_{21} and e_{22} in

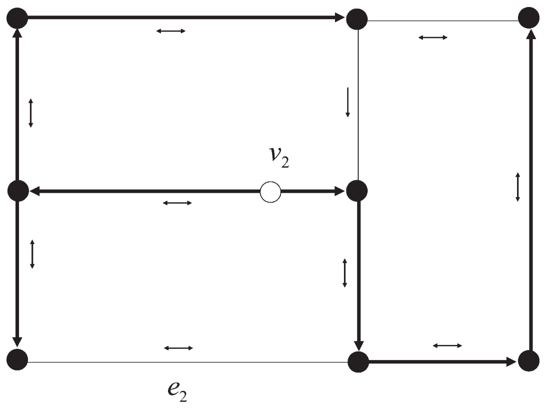

Figure 18. Shortest-path tree (the heavy line segments) rooted at the white circle (v_2) for the network indicated by the continuous line segments (arrows along the line segments indicate direction).

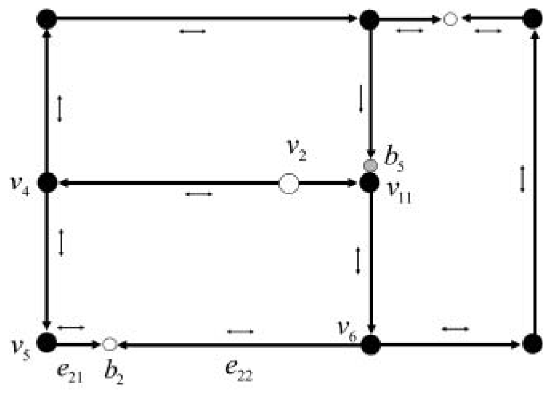

Figure 19. Extended shortest-path tree rooted at the white circle (v_2) for the network indicated by the continuous lines (the arrowed lines along the line segments show directions).

figure 19). The second type of auxiliary vertex is generated on the terminal vertex of every one-way uncovered edge (the grey circle in figure 19). This edge is treated as an open edge in the sense that the terminal vertex is not included in the uncovered one-way edge. If the break vertices, divided edges, open edges, and terminal vertices are added to the SPT, the resulting tree covers the whole network. This modified SPT is termed an *extended SPT*, abbreviated to ESPT. When specifying a root vertex v_i, ESPT(v_i) is used. A computational method for constructing the ESPT is developed in the next section.

3.2 *Directed ordinary N-VD*

A naïve method for constructing the directed ordinary N-VD is: first, to construct n_g ESPTs rooted at the n_g generator points; and second, to choose the nearest generator point at every point on the network using the n_g ESPTs. While such a method is possible (in fact it was used to construct the N-VDs in sections 3.4 and 3.5), a good trick can be used instead, which requires only one ESPT.

The trick is: first, add one dummy vertex p_0 (the broken line circle in figure 20) and n_g dummy edges that join p_0 and generator points p_i, $i=1,\ldots, n_g$ (the broken line

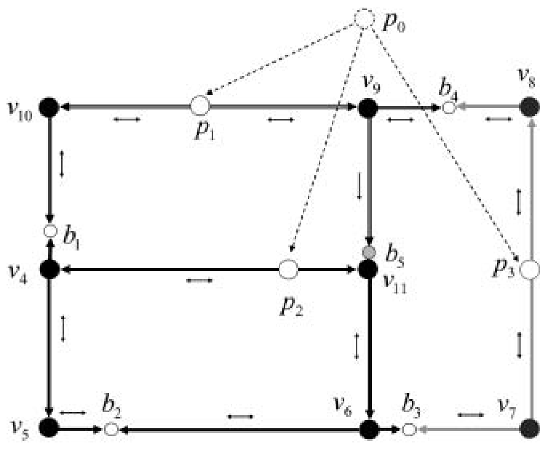

Figure 20. Directed ordinary N-VD consisting of ESPT*(p_1), ESPT*(p_2), and ESPT*(p_3) indicated by the double line segments, black line segments, and grey line segments, respectively (p_0 is a dummy vertex, the broken lines are dummy edges, p_1, p_2, and p_3 are generator points).

segments in figure 20) with zero edge length d_S (p_0, p_i)=0, to the original network; second, construct ESPT(p_0) rooted at the dummy vertex p_0; last, delete the dummy vertex and dummy edges from ESPT(p_0). Then, ESPT(p_0) is decomposed into n_g subtrees, ESPT*(p_1), … , ESPT*(p_{n_g}) (in figure 20, ESPT*(p_1), ESPT*(p_2) and ESPT*(p_3) are indicated by the double line segments, the black line segments and the grey line segments, respectively). The Voronoi subnetwork Vor$_i$ is given by ESPT*(p_i), and the directed ordinary N-VD, Vor, is written as Vor={ESPT*(p_1), …, ESPT*(p_{n_g})}.

Note that the method proposed by Erwig (2000) is similar to the above idea in principle, although he deals with a graph rather than a network. Erwig formulated a computational method, called the *parallel Dijkstra method*, for constructing the Voronoi diagram on a graph, called the *graph VD*.

Consider a network, N (V, E) consisting of the set of vertices $V=\{v_1,\ldots, v_n\}$, and the set of edges, $E=\{e_1,\ldots,e_{n_e}\}$ (figure 21(*a*)). An edge e_i may be one-way or two-way. The generator points, $P=\{p_1,\ldots,p_{n_g}\}$ (the large white circles in figure 21(*b*)), are placed on the edges of E (including the vertices of V), and the generator points are

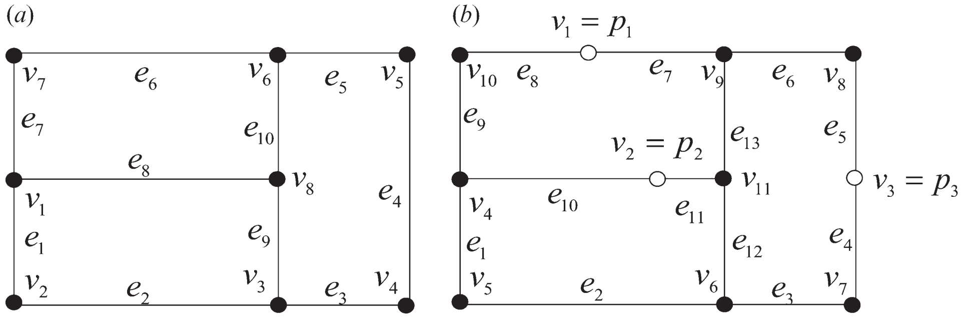

Figure 21. (*a*) Original network $N(V, E)$ and (*b*) its modified network $N(V^*, E^*)$ by inserting generator points $v_1=p_1$, $v_2=p_2$, $v_3=p_3$ (white circles) in the network $N(V, E)$.

regarded as vertices (figure 21(*b*)). The order of n_g is assumed to be less than or equal to that of n_v, i.e. $n_g \leqslant n_v$. In practice, n_g is usually much smaller than n_v.

Next, consider the set of vertices and generator points, i.e. $V^* = V \bigcup P$, and re-index the elements of $V \bigcup P$ as $V^* = \{v_1, \ldots, v_{n_v^*}\}$ (figure 21(*b*)). For convenience, the first n_g elements are generator points, i.e. $v_i = p_i$, $i = 1, \ldots, n_g$. Note that the relations $n_v \leq n_v^* \leq n_v + n_g$ hold. The added generator points divide the edges of E and produce new edges (e.g. e_4 in figure 21(*a*) is divided at v_3 into e_4 and e_5 in figure 21(*b*)). The set of resulting edges is denoted by E^*, and its elements are re-indexed as $E^* = \{e_1, \ldots, e_{n_e^*}\}$ (figure 21(*b*)). The numbers n_e, n_g and n_e^* satisfy the relations $n_e \leq n_e^* \leq 2n_g + (n_e - n_g) = n_g + n_e$. The sets V^* and E^* produce a new network, $N(V^*, E^*)$ (figure 21(*b*)), on which a directed ordinary N-VD is to be constructed.

Our computational method is similar to the label-setting method (Dijkstra 1959, Denardo and Fox 1979, Fredman and Tarjan 1987), but a few distinct modifications are made. Following the label-setting method, three types of labelled sets are used: the set of *permanently labelled vertices* $V_{P\text{–label}}$, the set of *temporarily labelled vertices* $V_{T\text{–label}}$ and the set of *unlabelled vertices* V_{Unlabel}. The first set, $V_{P\text{–label}}$, consists of vertices with nearest generator points that have been determined by the current stage of processing. The second set, $V_{T\text{–label}}$, consists of vertices with nearest generator points that are under consideration at the current stage of processing. The last set, V_{Unlabel}, consists of vertices with nearest generator points that have not yet been considered. During the processing period, the label of a vertex changes from an unlabelled vertex to a temporarily labelled vertex, and eventually to a permanently labelled vertex.

At every moment in processing, each vertex v_i is characterized by two sets and one scalar, $V_{\text{gene}}(v_i)$, $V_{\text{pred}}(v_i)$ and $d_{\text{gene}}(v_i)$. The first set $V_{\text{gene}}(v_i)$ is the set of tentative nearest generator points for v_i. Usually, the label-setting methods for constructing an SPT do not use $V_{\text{gene}}(v_i)$, because there is only one root vertex. When constructing N-VDs, more than one SPT is used, and so the information about root vertices (generator points) should be recorded. Usually, the information is treated with one variable for each v_i, but here a set is used, because in constructing the N-VD there are two cases where a set is useful. In the first case, a boundary point of two Voronoi subnetworks Vor_i and Vor_j happens to be on a vertex v_i of V and the vertex has two nearest generator points, say p_j and p_k, i.e. $V_{\text{gene}}(v_i), = \{p_j, p_k\}$. This case looks as if it would be rare, but we found in actual applications that such a case occurs more frequently than expected, especially when a network $N(V, E)$ is a grid street system, and the generator points of P are on the vertices of V. The second case occurs on all break vertices of an ESPT (this case is not a special case). Every boundary vertex has two nearest generator points. Therefore, a set, $V_{\text{gene}}(v_i)$, is used.

The second set $V_{\text{pred}}(v_i)$ is the set of predecessors to v_i, on the shortest paths from the tentative generator points in $V_{\text{gene}}(v_i)$, to v_i. $V_{\text{pred}}(v_i)$ is also a set, because a boundary point has different predecessors.

The distance $d_{\text{gene}}(v_i)$ indicates the shortest-path distance from a tentative nearest generator point in $V_{\text{gene}}(v_i)$ to v_i. In addition to the above terms, $V_{\text{adja}}(v_i)$ is used for indicating the set of vertices that are joined with outward edges from v_i (e.g. $V_{\text{adja}}(v_6) = \{v_5, v_8\}$ in figure 22) .

For consistent use of terms, the terms and notation introduced above are used in the following algorithms, but a few remarks are made from a computational viewpoint. In programming, the sets $V_{P\text{–label}}$, $V_{\text{gene}}(v_i)$, $V_{\text{pred}}(v_i)$, and $V_{\text{adja}}(v_i)$ are

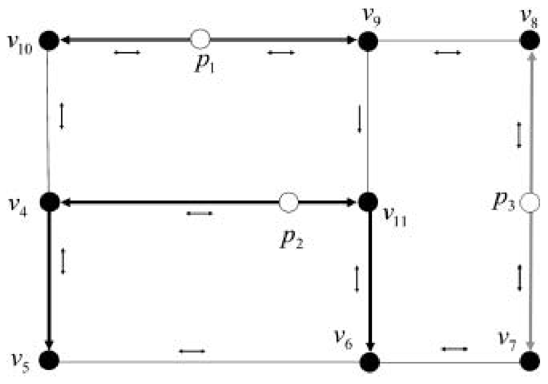

Figure 22. Shortest-path trees rooted at generator points p_1, p_2, and p_3, i.e. SPT*(p_1) (double lines), SPT*(p_2) (black lines) and SPT*(p_3) (grey lines), respectively.

defined as lists (Aho *et al.* 1974, pp. 44–47). The set $V_{T\text{–label}}$ is associated with d_{gene} (v_i), i.e. $\{ d_{\text{gene}}(v_i) | v_i \in V_{T-\text{label}}\}$, and their data are managed with a Fibonacci heap (Fredman and Tarjan 1987), which is subsumed under the priority cue. The operations using the Fibonacci heap are:

- INSERT(v_i): to insert a vertex v_i in $V_{T\text{–label}}$;
- DELETE(): to delete the vertex v_i ($=v_{\min}$) that has the minimum value among $\{d_{\text{gene}}(v_i)|v_i \in V_{T-\text{label}}\}$ from $V_{T\text{–label}}$;
- RESTRUCTURE(v_k): to restructure $V_{T\text{–label}}$ with v_k having a new value d_{gene} (v_k); and
- FIND($v_{\min}$): to find and return the vertex $v_{\min}$ having the minimum value d_{gene} ($v_{\min}$).

Because V_{Unlabel} can be implicitly treated in terms of $d_{\text{gene}}(v_i) = \infty$, it does not appear in the following algorithm. Note that the following algorithm deals with the outward N-VD, but the inward N-DV is also defined in a similar way.

The algorithm for constructing the directed ordinary N-VD has two phases. The first phase is to construct SPT*(p_1), . . . , SPT*(p_{n_g}), where SPT*(p_i) is the shortest-path tree rooted at p_i, i=1,…, n_g (figure 22); the second phase is to extend the trees to ESPT*(p_1), . . . , ESPT*(p_{n_g}) (figure 20).

Phase 1: construction of SPT*(p_1), . . . , SPT*(p_{n_g}).
Step 1: initialize.
For each $v_i(=p_i)$ i=1,…, n_g,

$$V_{\text{gene}}(v_i) := \{p_i\},\ V_{\text{pred}}(v_i) := \{p_0\}\ \text{(a dummy vertex)},\ d_{\text{gene}}(v_i) = 0,\ \text{and INSERT}(v_i).$$

For each, v_i, $i=n_g+1,\ldots,n_v^*$,

$$V_{\text{gene}}(v_i) := \{v_\infty\}\ \text{(a dummy vertex)},\ V_{\text{pred}}(v_i) := \{v_o\}\ \text{(a dummy vertex)},$$
$$\text{and } d_{\text{gene}}(v_i) := \infty\ V_{P-\text{label}} := \phi$$

Step 2: find the minimum vertex
FIND($v_{\min}$), DELETE() and $V_{P\text{–label}} := V_{P-\text{label}} \cup \{v_{\min}\}$.

Fix the values of $V_{\text{gene}}(v_{\min})$, $V_{\text{pred}}(v_{\min})$, $d_{\text{gene}}(v_{\min})$ permanently for $v_{\min}$.

Step 3: scan for outward edges incident to $v_{\min}$
For each $v_k \in V_{\text{adja}}(v_{\min}) \backslash V_{P\text{–label}}$, $D := d_{\text{gene}}(v_{\min}) + d_S(v_{\min}, v_k)$.
If $d_{\text{gene}}(v_k) = \infty$, then $V_{\text{gene}}(v_k) := V_{\text{gene}}(v_{\min})$, $V_{\text{pred}}(v_k) := \{v_{\min}\}$, $d_{\text{gene}}(v_k) := D$, and INSERT($v_k$).
If $d_{\text{gene}}(v_k) > D$, then $V_{\text{gene}}(v_k) := V_{\text{gene}}(v_{\min})$, $V_{\text{pred}}(v_k) := \{v_{\min}\}$, $d_{\text{gene}}(v_k) := D$, and RESTRUCTURE($v_k$).
If $d_{\text{gene}}(v_k) := D$, then $V_{\text{gene}}(v_k) := V_{\text{gene}}(v_k) \cup V_{\text{gene}}(v_{\min})$, $V_{\text{pred}}(v_k) := V_{\text{pred}}(v_k) \cup \{v_{\min}\}$.

Step 4: continue or terminate
If $V_{T\text{–label}} \neq \phi$, go to Step 2; otherwise, terminate.

Two remarks are made on the above algorithm. First, the ordinary label-setting method for an SPT has only one vertex in $V_{T\text{–label}}$ in Step 1, but the above algorithm has n_g vertices in Step 1. Second, the ordinary label-setting method treats only one generator point for a vertex, but the above algorithm treats more than one generator point.

3.2.1 Computation time in phase 1. In Step 1, each operation, $V_{\text{gene}}(v_i) :=$, $V_{\text{pred}}(v_i) :=$, $d_{\text{gene}}(v_i) :=$, $V_{P\text{–label}} :=$, and INSERT, requires a constant time, and these operations are carried out for all vertices of V^*. As a result, the computation time is $O(n_v^*)$.

In Step 2, our program employs the Fibonacci heap method (Fredman and Tarjan 1987). FIND requires O(1), and DELETE requires $O(\log n_v^*)$ (Fredman and Tarjan 1987). This operation is done for the vertices of V^*, and so the total computation time is $O(n_v^* \log n_v^*)$. The operations $V_{P\text{–label}} :=$and FIND are carried out for the vertices of V and so the computation time is $O(n_v)$.

In Step 3, the operation $V_{\text{gene}}(v_k) := V_{\text{gene}}(v_k) \cup V_{\text{gene}}(v_{\min})$ depends on the number of generator points in $V_{\text{gene}}(v_{\min})$. Thus, this operation requires computation time of order $O(n_g)$. Each of the other operations, $V_{\text{pred}}(v_i) :=$, and $d_{\text{pred}}(v_i) :=$, $V_{P\text{–label}} :=$, INSERT and RESTRUCTURE, requires a constant time. These operations are carried out for scanned edges joining the adjacent vertices (i.e. $v_k \in V_{\text{adja}}(v_{\min}) \backslash V_{P\text{–label}}$). Therefore, the total computation time is dominated by the first operation, i.e. it is $O(n_g\, n_e^*)$.

In Step 4, the decision requires a constant time, and the number of the decisions is $O(n_v)$.

Summing up, the total computation time is $O(n_g\, n_e^* + n_v^* \log n_v^*)$ in theory, but when n_g is much smaller than n_e^* and n_v^*, which often happens in practice, it is dominated by $O(n_e^* + n_v^* \log n_v^*)$.

The second phase is to construct ESPT*(p_1), . . . , ESPT*(p_{n_g}) from SPT*(p_1), . . . , SPT*(p_{n_g}) obtained in the first phase. Consider the set of uncovered edges that are one-way and the set of uncovered edges that are two-way. $^+(e_i)$ and $^-(e_i)$ denote the end vertices of a two-way edge e_i (+ and – are arbitrary), $^+(e_i)$ denotes the start vertex, and $^-(e_i)$ denotes the end vertex of a one-way edge e_i.

Phase 2: construction of the extended shortest-path trees ESPT*(p_1), . . . , ESPT*(p_{n_g}).
Step 1: find the edges that are not covered with the shortest-path trees SPT*(p_1), . . . , SPT*(p_{n_g}) (e.g. the hair lines in figure 22).

For every two-way edge e_i in E^*, if $|d_{\text{gene}}(\partial^+ e_i) - d_{\text{gene}}(\partial^- e_i)| \neq d_{\text{S}}(\partial^+ e_i, \partial^- e_i)$, $d_{\text{gene}}(\partial^+ e_i) \neq \infty$, and $d_{\text{gene}}(\partial^- e_i) \neq \infty$, then e_i is an uncovered edge; otherwise it is a covered edge.
For every one-way edge e_i in E^*, if $|d_{\text{gene}}(\partial^+ e_i) - d_{\text{gene}}(\partial^- e_i)| \neq d_{\text{S}}(\partial^+ e_i, \partial^- e_i)$, $d_{\text{gene}}(\partial^+ e_i) \neq \infty$, and $d_{\text{gene}}(\partial^- e_i) \neq \infty$, then e_i is an uncovered edge; otherwise it is a covered edge.

Step 2: generate break vertices (e.g. the small white circles in figure 20).
For each uncovered edge e_i, if it is a two-way edge, determine the location of a break vertex b_i by solving the following equations with respect to unknown $d_S\,({}^+e_i, b_i)$ and $d_S({}^-e_i, b_i)$ given that $d_{\text{gene}}({}^+e_i)$, $d_{\text{gene}}({}^-e_i)$ and $d_{\text{S}}\,({}^+e_i, {}^-e_i)$ are known:

$$d_{\text{gene}}(\partial^+ e_i) + d_{\text{S}}(\partial^+ e_i, b_i) = d_{\text{gene}}(\partial^- e_i) + d_{\text{S}}(\partial^- e_i, b_i)$$

$$d_{\text{S}}(\partial^+ e_i, b_i) + d_{\text{S}}(\partial^- e_i, b_i) = d_{\text{S}}(\partial^+ e_i, \partial^- e_i)$$

If e_i is a one-way edge, generate a vertex b_i at ${}^-(e_i)$.
Step 3: set values
For the vertex b_i on a two-way edge:

$$V_{\text{gene}}(b_i) := V_{\text{gene}}(\partial^+ e_i) \cup V_{\text{gene}}(\partial^- e_i), \;\; V_{\text{pre}}(b_i) := \{\partial^+ e_i, \partial^- e_i\}$$

$$d_{\text{gene}}(b_i) := d_{\text{gene}}(\partial^+ e_i) + d_{\text{S}}(\partial^+ e_i, b_i)$$

For the vertex b_i on a one-way edge:

$$V_{\text{gene}}(b_i) := V_{\text{gene}}(\partial^+ e_i), \;\; V_{pre}(b_i) := \{\partial^+ e_i\}$$

$$d_{\text{gene}}(b_i) := d_{\text{gene}}(\partial^+ e_i) + d_{\text{S}}(\partial^+ e_i, b_i)$$

3.2.2 Computation time in phase 2. In Step 1, the 'if' judgement of O(1) is done n_e^* times, and so the computation time is O(n_e^*).

In Step 2, solving the equations requires a constant time, and this is done for edges. Therefore, the time is O(n_e^*).

In Step 3, each operation is achieved in a constant time and the operations are done for break vertices on uncovered edges, in time O(n_e^*).

Summing up the computation times in phases 1 and 2, the total computation time for constructing the directed ordinary N-VD is O($n_g\, n_e^* + n_v^* \log n_v^*$), but in practice, n_g is much smaller, and so the time is dominated by O($n_e^* + n_v^* \log n_v^*$).

3.3 *Additively weighted N-VD*

Equation (5) shows that $d_{\text{AW}}(p_i, p_i) = d_{\text{S}}(p_i, p_i) + \beta_i = \beta_i$. This implies that in the context of ESPT(p_0) shown in figure 20, the shortest-path distance from the dummy vertex p_0 to a generator point p_i is β_i (the broken line segments in figure 20). Therefore, the algorithm is the same as that in section 3.2 except for replacing the second line in Step 1 of Phase 1 with

$V_{\text{gene}}(v_i) := \{p_i\}$, $V_{\text{pred}}(v_i) := p_0$ (a dummy vertex), $d_{\text{gene}}(v_i) := \beta_i$ and INSERT(v_i)

This change does not affect computation complexity, and so the computation time for constructing the (directed) additively weighted N-VD is the same as that for the directed ordinary N-VD, i.e. $O(n_e^* + n_v^* \log n_v^*)$.

3.4 *Multiplicatively weighed N-VD*

A computational method for constructing the multiplicatively weighted N-VD is greatly different from the directed ordinary N-VD and the additively weighted N-VD shown in the above, because the trick shown in figure 20 does not work for the multiplicatively weighted N-VD: $d_{\mathrm{MW}}(p, p_i) = \alpha_i\, d_{\mathrm{S}}(p, p_i)$ cannot be treated in terms of the length of dummy edges. This section proposes an alternative method. Note that this method deals with non-directed networks, but a slight modification of this method can deal with directed networks.

Step 1: construct the extended shortest-path tree $\mathrm{ESPT}(p_i)$
For a given non-directed network, $N(V^*, E^*)$, assign the value $\alpha_i d_{\mathrm{S}}(^+e_j, {}^-e_j)$ to edge e_j as its edge length, and denote the network with these attribute values by $N(V^*, E^*; \alpha_i)$ (in this term, the original network $N(V^*, E^*)$ is written as $N(V^*, E^*; 1)$). Construct $\mathrm{ESPT}(p_i)$ for $N(V^*, E^*; \alpha_i)$, $i = 1, \ldots, n_g$ (e.g. $\mathrm{ESPT}(p_1)$ and $\mathrm{ESPT}(p_2)$ shown in figure 23(*a*) and (*b*); n_g ESPTs are generated). Denote the set of the break vertices of $\mathrm{ESPT}(p_i)$ by B_i, and $B = \bigcup_{i=1}^{n_g} B_i$ (the small white circles in figure 23(*c*)).

Step 2: construct a modified network – I
Construct a modified network $N(V_B^*, E_B^*)$ by adding the vertices of B to the edges of E^*, and dividing the edges at the break vertices. The set of resulting edges is denoted by E_B^* and $V_B^* = V^* \cup B$ (figure 23(*c*)).

Step 3: obtain the lower-bound envelope
For each edge e_i of E_B^*, obtain the lower-bound envelope function, $f_{Li}(x)$, of the distance functions $f_{i1}(x), \ldots, f_{in_g}(x)$, i.e. $f_{Li}(x) = \min\{f_{i1}(x), \ldots, f_{in_g}(x)\}$ (the heavy line segments in figure 24), where $f_{ij}(x)$ indicates the shortest-path distance from the generator point p_j to a point p at $x = d_{\mathrm{S}}(^+e_i, p)$ on e_i, i.e.

$$f_{ij}(x) = \delta_j \alpha_j x + \alpha_j d_{\mathrm{S}}(p_j, \partial^+ e_i), \;\; j = 1, \ldots, n_g$$

$$\delta_j = 1 \text{ if } d_{\mathrm{S}}(p_j, \partial^+ e_i) \le d_{\mathrm{S}}(p_j, \partial^- e_i); \text{ otherwise, } \delta_j = -1$$

Note that $d_{\mathrm{S}}\,(p_i, {}^+e_i)$ and $d_{\mathrm{S}}\,(p_i, {}^-e_i)$ are obtained from the $\mathrm{ESPT}(p_j)$ in Step 1; the function is linearly increasing or decreasing; a peak is at either end-point, and there

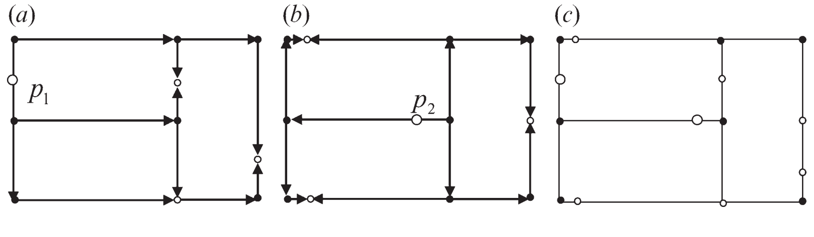

Figure 23. Extended shortest-path tree $\mathrm{ESPT}(p_1)$ rooted at p_1 (*a*), the extended shortest-path tree $\mathrm{ESPT}(p_2)$ rooted at p_2 and the modified network by adding the break points (small circles in (*a*) and (*b*)) to the original network$N(V_B^*, E_B^*)$.

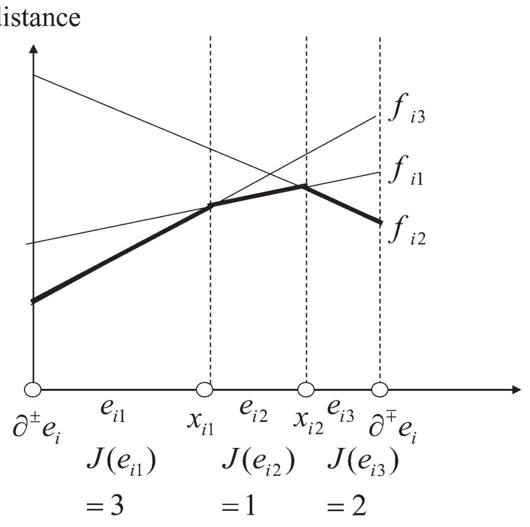

Figure 24. Lower-bound envelope.

can be no peak between the end-points, as shown in figure 24; and there are n_g distance functions for each edge. Note that $f_{Li}(x)$ is associated with the data of the intersection point x_{im} of the two distance functions that form the lower-bound envelope (e.g. x_{i1}, x_{i2} in figure 24), and the data of the suffix h of the distance function f_{ih} (x), $x \in e_{ij}$, denoted by $J(e_{ij})$, that forms the lower-bound envelope on an edge e_{ij} (e.g. $J(e_{i1})=3$).

Step 4: construct a modified network—II
Construct a modified network by dividing each edge e_i at $x_{i1},\ldots,x_{ik_i-1}$ and denote the resulting edges by $e_{i1},\ldots,e_{ik_i}$ (figure 24). The multiplicatively weighted Voronoi subnetwork Vor_h of the generator point p_h is identified by

$$\mathrm{Vor}_h = \left\{ e_{ij} \middle| \, J(e_{ij}) = h, \, i = 1,\ldots,n^*_{e_g}, \, j = 1,\ldots,k_i \right\}$$

3.4.1 Computation time. In Step 1, the ESPT is constructed n_g times. The construction of one ESPT is, as shown in section 3.2, carried out in $O(n^*_e + n^*_v \log n^*_v)$. Therefore, the total computation time is $O(n_g\,(n^*_e + n^*_v \log n^*_v))$.

In Step 2, the number of break vertices in B_i is $O(n^*_e)$ for each ESPT(p_i). Thus, the number of break vertices in B is $O(n_g n^*_e)$. Because the network $N(V^*_B, E^*_B)$ is constructed by inserting vertices of B on the edges of E^*, the computation time for this construction is $O(n_g n^*_e)$.

In Step 3, the lower-bound envelope for each edge of E^*_B is constructed with the divide-and-conquer method using the Davenport–Schinzel sequence (section 6.2 in Sharir and Agarwal 1995). This computation time is proved to be $O(n_g \log^* n_g \log n_g)$, where $\log^* n_g$ means taking logs the number of times that gives the first negative value, and $n_g \log^* n_g$ is almost linear with respect to n_g. Therefore, $O(n_g \log^* n_g \log n_g) \approx O(n_g \log n_g)$. The number of edges in E^*_B is $O(n_g n^*_e)$. In total, therefore, the computation time in Step 3 is $O(n_g^2 n^*_e \log n_g)$.

In Step 4, the computation time is proportional to the number of vertices of B, with order $O(n_g n_e^*)$.

Summing up the above computation times, the computation time for constructing the multiplicatively weighted N-VD is $O(n_g^2 n_e^* \log n_g + n_g n_v^* \log n_v^*)$, which is dominated by the first term.

3.5 *kth-nearest-point N-VD*

The computational method for constructing the kth-nearest-point N-VD proposed in this section is similar to that for the multiplicatively weighted N-VD. Stated precisely, Step 1 is the same except that $\alpha_i=0$ for all $i=1,\ldots, n_g$. Step 2 is the same. Step 3 is replaced with the following.

Step 3: obtain the kth-lower-bound envelope
For each edge e_i of E_B^*, obtain the kth-lower-bound envelope function, $f_{L(k)i}(x)$, of the distance functions, i.e. $f_{L(k)i}(x)=$ kth minimum $\{f_{i1}(x),\ldots,f_{in_g}(x)\}$ (the example for $k=3$ is indicated by the heavy line segments in figure 25).

Step 4 is the same.

3.5.1 Computation time. The computation time of each step is the same as that for the multiplicatively weighted N-VD, except for the computation in Step 3. A method for calculating the kth-lower-bound envelope is outlined as follows. First, order $f_{i1}(0),\ldots,f_{in_g}(0)$ from the smallest to the n_g th smallest (the largest), denote the resulting ordered functions by $f_{i(1)}(0),\ldots,f_{i(n_g)}(0)$, and choose the kth function $f_{i(k)}(x)$ (in the example of figure 25, $f_{i(3)}(x)$). If the function $f_{i(k)}(x)$ goes up (down), then calculate the intersection point x_{i1} between $f_{i(k)}(x)$ and the first nearest function that goes down (up), denoted by $f_{i(k\oplus 1)}(x)$ (in figure 15, $f_{i(3\oplus 1)}(x)=f_{i(5)}(x)$). Next, calculate the intersection point x_{i2} between $f_{i(3\oplus 1)}(x)$ and the first nearest function that goes up (down), denoted by $f_{i(k\oplus 2)}(x)$ (in figure 25, $f_{i(3\oplus 2)}(x)=f_{i(1)}(x)$), and so on. The computation of ordering $f_{i1}(0),\ldots,f_{in_g}(0)$ requires $O(n_g \log n_g)$ time. The intersection point is obtained in constant time, and the number of intersection points is possibly n_g. Therefore, the computation time in Step 3 is dominated by $O(n_g \log n_g)$. Summing

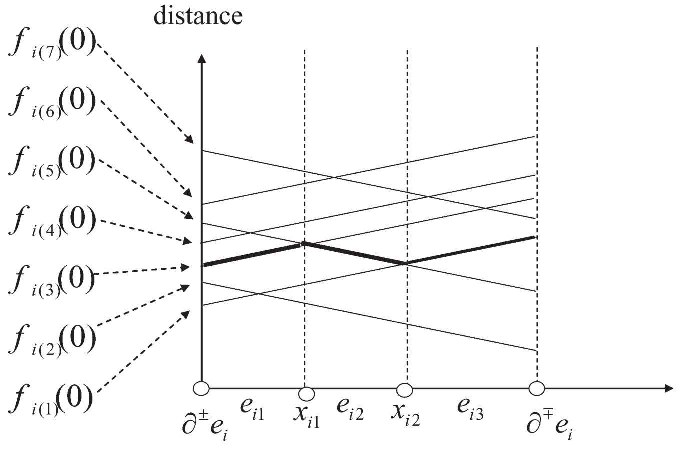

Figure 25. kth-lower-bound envelope.

up the computation times in Steps 1–4, the time for computing the kth-nearest N-VD is $O(n_g^2 n_e^* \log n_g + n_g n_v^* \log n_v^*)$, which is dominated by the first term.

3.6 *Line N-VD*

Although points and lines are different in geometrical form, the method for constructing the ordinary N-VD in section 3.2 is applicable to that for constructing the line N-VD. As mentioned in section 2.4, generator lines on a network $N(V, E)$ are given by subnetworks (termed *generator subnetworks*) $N(V_{S1}, E_{S1}),\ldots, N(V_{Sm}, E_{Sm})$ of the network $N(V,E)$, where $V_{Si}=\{v_{Si1},\ldots,v_{Sim_{vi}}\}$, $v_{Sik}\in V$ and $E_{Si}=\{e_{Si1},\ldots,e_{Sim_{ei}}\}$, $e_{Sik}\in E$ (the heavy line segments with their end-points in figure 26). As for the ordinary N-VD and the additively weighted N-VD in sections 3.2 and 3.3, a similar trick is applicable, but setting the dummy vertices and edges during initialization is different. First, a dummy vertex p_{0i} (the heavy broken circles in figure 26) is added for each generator subnetwork $N(V_{Si},E_{Si})$, and m_{vi} dummy edges, E_{0i}, are generated by joining p_{0i} and v_{Sij}, $j=1,\ldots,m_{vi}$ (the heavy broken line segments in figure 26). The length of every dummy edge is zero, and the length of every edge in E_{Si}, $i=1,\ldots, m$ is replaced with zero. Second, one dummy vertex p_0 is added (the light broken line circle in figure 26), and m dummy edges, E_0, are generated by joining p_0 and p_{0i}, $i=1,\ldots, m$ (the light broken line segments in figure 26). Third, the ESTP(p_0) is constructed for the modified network $N(V\cup\{p_0,p_{01},\ldots,p_{0m}\},E\cup E_{S1}\cup\cdots\cup E_{Sm}\cup E_0)$. Fourth, the dummy vertex p_0 and the dummy edges of E_0 are deleted from the ESTP(p_0). As a result, m subtrees ESPT*(p_{01}), . . . , ESPT*(p_{0m}) are obtained. The ith Voronoi subnetwork Vor$_i$ of the ith generator subnetwork $N(V_{Si}, E_{Si})$ is given by the subnetwork that is obtained by deleting the dummy node p_{0i} and the dummy edges of E_{0i} from ESTP*(p_{0i}).

In computational terms, the algorithm for constructing the line N-VD is given by that for the ordinary N-VD in section 3.2, but Step 1 of Phase 1 is replaced with:

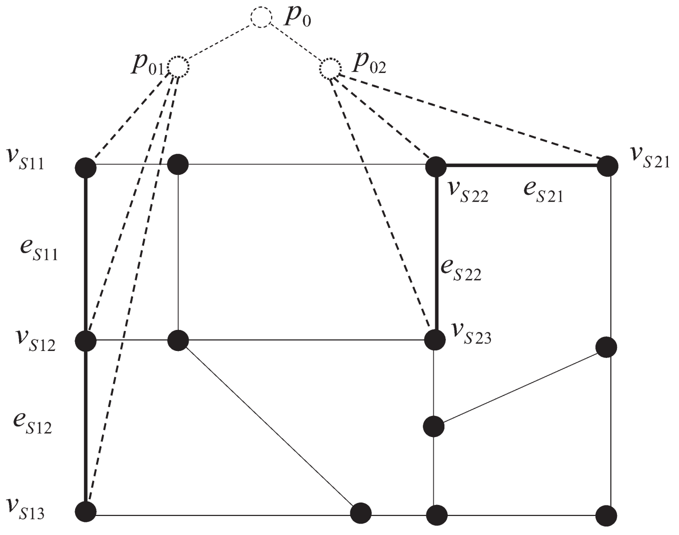

Figure 26. Dummy vertices p_0, p_{01}, p_{02} and dummy edges (the broken lines) for constructing a line N-VD.

Step 1: initialize
For each $v_{Sij} \in V_{Si}$, $i=1,\ldots, m$,

$$V_{\text{gene}}(v_{Sij}): =\{v_{Sij}\},\ V_{\text{pred}}(v_{Sij}): =\{p_{0i}\},\ d_{\text{gene}}(v_{Sij}) : =0 \text{ and INSERT } (v_{Sij})$$

For each $e_j \in E_{Si}$, $i=1,\ldots, m$,

$$d_S(\partial^+ e_j, \partial^- e_j) : =0$$

For each $v_j \in V \setminus \bigcup_{i=1}^{m} V_{Si}$,

$$V_{\text{gene}}(v_j) : =\{v_\infty\},\ V_{\text{pred}}(v_j) : =\{v_0\},\ d_{\text{gene}}(v_j) : =\infty,\ V_{P-\text{label}} : =\phi$$

3.6.1 Computation time. The above change does not affect the computation time, but the expression is slightly different, because the vertices in $\bigcup_{i=1}^{m} V_{Si}$ are elements of V. The computation time for constructing the line N-VD is $O(n_e+n_v \log n_v)$.

3.7 *Polygon N-VD*

As noted in section 2.5, because a polygon in the network context is represented by a loop (a subnetwork), the method for constructing the polygon N-VD is exactly the same as that for the line N-VD. Therefore, the computation time is the same: $O(n_e+n_v \log n_v)$.

3.8 *Point-set N-VD*

The point-set N-VD can be constructed by the method for the line N-VD if the set of generator point sets $P=\{P_1,\ldots, P_n\}$ is transformed into the set of fictitious generator subnetworks. For a given generator point set $P_i=\{p_{i1},\ldots,p_{im_i}\}$, the set of edges $E_{Si}=\{e_{i1},\ldots,e_{im_i-1}\}$, are generated ($i=1,\ldots n$), where e_{ij} is the line segment joining p_{ij} and p_{ij+1} (the broken line segments in figure 27). Then, the vertex set $V_{Si}=P_i$ and the edge set E_{Si} form a subnetwork, $N(V_{Si},E_{Si})$, and this subnetwork can be regarded as

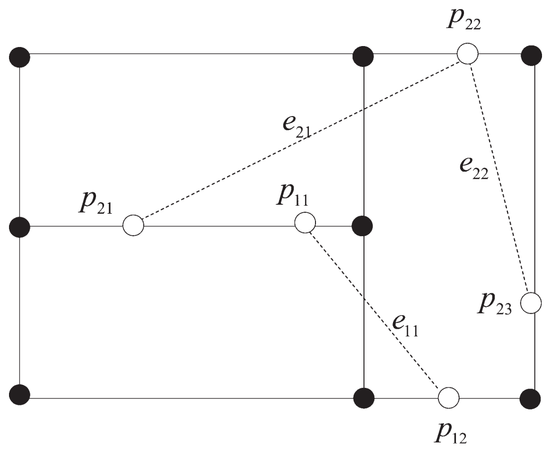

Figure 27. Dummy edges (the broken lines) for point sets.

a generator subnetwork for the line N-VD. Once this transformation is made, the point-set N-VD can be treated as the line N-VD, and the construction of the point-set N-VD is exactly the same as that of the line N-VD.

3.8.1 Computation time. The number n_v^{**} of vertices is given by $n_v \leqslant n_v^{**} \leq n_v + \sum_{i=1}^{n} m_i$, and the number of edges is given by $n_e \leqslant n_e^{**} \leq n_e + \sum_{i=1}^{n} m_i$. Therefore, the computation time is $O(n_e^{**} + n_v^{**} \log n_v^{**})$.

4. GIS-based tools for constructing generalized network Voronoi diagrams

N-VDs will be used not only by professional geographical information scientists but also by non-professional users, such as researchers in criminology and ecology, and managers of retail stores, who cannot spend much time in computer programming and wish to have user-friendly GIS-based tools for generalized N-VDs. To respond to such demand, we are developing GIS-based tools for generalized N-VDs, which are or will be included in the toolbox for Spatial Analysis on a NETwork, named SANET (Okabe *et al.* 2006b, c). At present, Version 3 is available, which includes the ordinary N-VD and additively weighted N-VD. The computer programs for the directed N-VD, line N-VD, polygon N-VD and point-set N-VD have been developed, and will be included in version 4 of SANET. The diagrams of the generalized N-VDs shown in section 3 were drawn by these tools. SANET can be downloaded from http://okabe.t.u-tokyo.ac.jp/okabelab/atsu/sanet/sanet-index2.html without charge for non-profit users.

5. Conclusion

In urbanized areas, as shown in figure 2, it is difficult to approximate travel distance by Euclidean distance. In particular, if the Euclidean distance is less than 500, the Euclidean distance and the corresponding shortest-path distance are significantly different. Because the radii of many service facilities in a city are less than 500 m (recall table 1), the service areas of the facilities cannot be well represented by the planar Voronoi diagram (P-VD). To overcome this limitation, this paper formulated Voronoi diagrams defined on a network, termed *network Voronoi diagrams* (N-VDs); specifically, six types of N-VDs: directed (inward/outward) N-VDs (section 2.1), weighted N-VDs (section 2.2), *k*th nearest point N-VDs (section 2.3), N-VDs generated by lines embedded on a network (section 2.4), N-VDs generated by polygons embedded on a network (section 2.5) and N-VDs generated by sets of points placed on a network (section 2.6).

The computational methods for constructing the generalized P-VDs in the literature cannot be applied to constructing the generalized N-VDs. Therefore, this paper has provided newly developed efficient computational methods for constructing the generalized N-VDs, using the extended shortest-path trees (section 3). It is notable that the computational method for constructing the directed N-VDs (section 3.2) is commonly applicable to those for the additively weighted N-VD (section 3.3), N-VDs generated by lines (section 3.6), polygons (section 3.7), and sets of points (section 3.8) with slight modifications.

The computational methods proposed in section 3 are not always easy to program. Considering that analysis with VDs is often used by computer-oriented researchers but also non-computer oriented users, this paper has developed user-friendly tools for constructing the generalized N-VDs, which were included or will be included in the software package called SANET (Spatial Analysis on a NETwork

(Okabe *et al.* 2006b, c), free for non-profit uses). It is hoped that the tools in section 4 will enable researchers to easily carry out spatial analysis with the generalized network Voronoi diagrams.

Acknowledgements

We express our thanks for anonymous referees for their valuable comments on an earlier draft.

References

AHO, A., HOPCROFT, J. and ULLMAN, J., 1974, *The Design and Analysis of Computer Algorithms* (Reading, MA: Addison-Wesley).

ALMEIDA, V.T. and GÜTING, R.H., 2006, Using Dijkstra's algorithm to incrementally find the *k*-nearest neighbors in spatial network databases. In *SAC'06*, April, Dijon, France, pp. 58–62.

ANSELIN, L., COHEN, J., COOK, D., GORR, W. and TITA, G., 2000, Spatial analyses of crime. In D. Duffee (Ed.), *Criminal Justice 2000: Volume 4. Measurement and Analysis of Crime and Justice*, pp. 213–262 (Washington, DC: National Institute of Justice).

BASHORE, T., TZILKOWSKI, W. and BELLIS, E., 1985, Analysis of deer–vehicle collision sites in Pennsylvania. *Journal of Wildlife Management*, **49**, pp. 869–774.

BOOTS, B., 1980, Weighting Thiessen polygons. *Economic Geography*, **56**, pp. 248–259.

BOWERS, K. and HIRSCHFIELD, A., 1999, Exploring links between crime and disadvantage in north-west England: an analysis using geographical information systems. *International Journal of Geographical Information Science*, **13**, pp. 159–184.

CLEVENGER, A.P., CHRUSZCZ, B. and GUNSON, K.E., 2003, Spatial patterns and factors influencing small vertebrate fauna road-kill aggregations. *Biological Conservation*, **109**, pp. 15–26.

ERWIG, M., 2000, The graph Voronoi diagram with applications. *Networks*, **36**, pp. 156–163.

DENARDO, E.V. and FOX, B.L., 1979, Shortest-route methods: 1. Reaching, pruning and buckets. *Operations Research*, **27**, pp. 161–186.

DIJKSTRA, E., 1959, A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, pp. 269–271.

FREDMAN, M.L. and TARJAN, R.E., 1987, Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association for Computing Machinery*, **34**, pp. 596–615.

FURUTA, T., SUZUKI, A. and INAKAWA, K., 2005, The *k*th nearest network Voronoi diagram and its application to districting problem of ambulance systems. *Discussion Paper*, No.0501, Center for Management Studies, Nanzan University.

JONES, A.P., LANGFORD, I.H. and BETHAM, G., 1996, The application of K-function analysis to the geographical distribution of road traffic accident outcomes in Norfolk, England. *Social Science and Medicine*, **42**, pp. 879–885.

KOLAHDOUZAN, M.R. and SHAHABI, C., 2005, Alternative solutions for continuous K nearest neighbor queries in spatial network databases. *GeoInformatica*, **9**, pp. 321–341.

LEVINE, N., KIM, K.E. and NITZ, L.H., 1995, Spatial analysis of Honolulu motor vehicle crashes: I. Spatial patterns. *Accident Analysis and Prevention*, **27**, pp. 663–674.

MAKI, N. and OKABE, 2005, A Spatio-temporal analysis of aged members of a fitness club in a suburb. *Proceedings of the Geographical Information Systems Association*, **14**, pp. 29–34.

MALLICK, S.A., HOCKING, G.J. and DRIESSEN, M.M., 1998, Road-kills of the eastern barred bandicoot (*Perameles gunnii*) in Tasmania: an index of abundance. *Wildlife Research*, **25**, pp. 139–145.

MCGUIGAN, D.R.D., 1981, The use of relationships between road accidents and traffic flow in 'black-spot' identification. *Traffic Engineering and Control*, **22**, pp. 448–453.

Miller, H.J., 1994, Market area delimitation within networks using geographic information systems. *Geographical Systems*, **1**, pp. 157–173.

Morita, M., Okunuki, K. and Okabe, A., 2001, A market area analysis on a network using GIS—A case study of retail stores in Nisshin city. *Papers and Proceedings of the Geographic Information Systems Association*, **10**, pp. 45–50.

Nicholson, A.J., 1989, Accident clustering: Some Simple Measures. *Traffic Engineering and Control*, **30**, pp. 241–246.

O'Driscoll, R.L., 1998, Descriptions of spatial pattern in seabird distributions along line transects using neighbor K statistics. *Marine Ecology Progress Series*, **165**, pp. 81–94.

Okabe, A., Boots, B., Sugihara, K. and Chiu, S.N., 2000, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd edition (Chichester, UK: Wiley).

Okabe, A. and Kitamura, M., 1996, A computational method for market area analysis on a network. *Geographical Analysis*, **28**, pp. 330–349.

Okabe, A. and Okunuki, K., 2001, A computational method for estimating the demand of retail stores on a street network using GIS. *Transactions in GIS*, **5**, pp. 209–220.

Okabe, A., Boots, B. and Satoh, T., 2006a, A class of local and global K-functions and cross K-functions. In *2006 Annual Meeting of the AAG*, Abstract ID=6898, March 2006, Chicago (in Japanese).

Okabe, A., Okunuki, K. and Shiode, S., 2006b, SANET: A toolbox for spatial analysis on a network. *Geographical Analysis*, **38**, pp. 57–66 (in Japanese).

Okabe, A., Okunuki, K. and Shiode, S., 2006c, The SANET toolbox: new methods for network spatial analysis. *Transactions in GIS*, **10**, pp. 535–550.

Okabe, A. and Yamada, I., 2001, The *K*-function method on a network and its computational implementation. *Geographical Analysis*, **33**, pp. 271–290.

Okabe, A., Yomono, H. and Kitamura, M., 1995, Statistical analysis of the distribution of points on a network. *Geographical Analysis*, 27, pp. 152–175.

Painter, K., 1994, The impact of lighting on crime, fear, and pedestrian street use. *Security Journal*, **5**, pp. 116–124.

Ratcliffe, H.J., 2002, Aoristic signatures and the spatio-temporal analysis of high volume crime patterns. *Journal of Quantitative Criminology*, **18**, pp. 23–43.

Ratcliffe, J.H. and McCullagh, M.J., 1999, Hotbeds of crime and the search for spatial accuracy. *Journal of Geographical Systems*, **1**, pp. 385–398.

Rau, K., 1841, Report to the Bulletin of the Belgian Royal Society. In *Precursors in Mathematical Economics: An Anthropology*, W. Baumol and S. Goldfield (Eds), pp. 181–182 (London: London School of Economics and Political Science).

Saeki, M. and MacDonald, D.W., 2004, The effects of traffic on the raccoon dog (*Nyctereutes procyonoides viverrinus*) and other mammals in Japan, *Biological Conservation*, **118**, pp. 559–571.

Satoh, T. and Okabe, A., 2006a, A tool development for spatial analysis with network Voronoi diagrams generated by lines and polygons. *GIS: Theory and Applications*, **14**, pp. 53–62.

Satoh, T. and Okabe, A., 2006b, Development of a tool for network Voronoi cross K function methods. *Discussion Paper No. 82*, Center for Spatial Infomation Science at the University of Tokyo.

Sharir, M. and Agarwal, P.K., 1995, *Davenport–Schinzel Sequences and Their Geometric Applications* (Cambridge: Cambridge University Press).

Shieh, Y.N., 1985, K.H. Rau and the economic law of market areas. *Journal of Regional Science*, **25**, pp. 191–199.

Snow, J., 1855, *On the Mode of Communication of Cholera*, 2nd edition (London: Churchill Livingstone), pp. 33–55.

Spooner, P.G., Lunt, I.D., Okabe, A. and Shiode, S., 2004, Spatial analysis of roadside Acacia populations on a road network using the network K-function. *Landscape Ecology*, **19**, pp. 491–499.

Yamada, Y. and Thill, J.-C., 2004, Comparison of planar and network K-functions in traffic accident analysis. *Journal of Transport Geography*, **12**, pp. 149–158.

Voronoi, G., 1908, Nouvelles applications des parametres continus a la theorie des formes quadratiques. Deuxieme Memoire: Recherches sur les parallelloedres primitifs. *Journal fur die Reine und Angewandte Mathmatik*, **133**, pp. 97–178.