CSIS Discussion Paper #26

CSIS Discussion Paper #26

The *K*-Function Method on a Network and its Computational Implementation

Atsuyuki OKABE* and Ikuho YAMADA**

April 20, 2000

* Center for Spatial Information Science, University of Tokyo

** Department of Urban Engineering, University of Tokyo

7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656 Japan

E-mail: atsu@ua.t.u-tokyo.ac.jp, iku@ua.t.u-tokyo.ac.jp

Acknowledgement

The authors thank Yukio Sadahiro and Kei-ichi Okunuki for their valuable comments on earlier versions of this paper. We acknowledge the support of the Ministry of Education Grant No.1480092.

This paper is a completely revised version of CSIS Discussion Paper Series No.23, "The K-Function Methods on a Network and their Computational Method" written in Japanese.

Abstract

This paper proposes two statistical methods, called the network K-function method and the network cross K-function method, for analyzing the distribution of points on a network. The former method is used for testing the hypothesis that points are uniformly and independently distributed over a network. The latter method is used for testing the hypothesis that points of one kind and points of another kind are independently distributed over a network. The paper first derives the equations of these functions. Second, these equations are implemented in computational procedures. Third, the computational orders of these procedures are obtained: the order of the K-function method is $O(n_Q^2 \log n_Q)$ and that for the network cross K-function is $O(n_Q \log n_Q)$, where n_Q is the number of nodes of a network.

1 Introduction

Among many kinds of statistical methods that examine the distribution of points on a plane, probably the K-function method (Ripley 1981) is one of the most frequently used methods in the literature (Cressie 1991; Fotheringham and Rogerson 1994; Bailey and Gatrell 1995). Like most statistical methods, the K-function method assumes a continuous plane with the Euclidean distance. Recently, however, Miller (1994) criticized the limitation of this assumption in conjunction with an increasing demand for detailed spatial analysis enhanced by geographic information systems (GIS). Actually this limitation is problematic when we apply the ordinary K-function method to a fairly small district. For instance, consider the location analysis of fast-food stands in a downtown. The stands cannot be located freely in the downtown, but they are located along streets. Consumers cannot fly to the stands, but they have to access to the stands through streets. The Euclidean distance is quite different from the route distance there. In addition to the discrepancy between the Euclidean plane and an actual space, there is a great demand for marketing in a small area with detailed spatial features, called micro-marketing (Buxton 1992). This discrepancy and the demand for micro-marketing require the development of a K-function method defined on a discrete network.

At present, however, this development is difficult because of the following reasons. First, the ordinary K-function method assumes an infinite homogeneous plane, but it is difficult to consider an infinite homogeneous network; a given network is almost always irregular and it is difficult to expand this irregular network to an infinite irregular network. Second, a method for computing a K-function on a network is much more complicated than the ordinary K-function method, because a network is usually inhomogeneous (note that the ordinary K-function assumes a homogeneous plane). Third, it is difficult to obtain and manage digital network data. Recently, the progress in GIS has overcome the third difficulty, but the first and second difficulties still remain unsolved. The objective of this paper is to overcome these difficulties and propose a practical K-function method defined on a network.

Although we hardly find a K-function method in the literature, we can find some statistical and/or computational methods for analyzing point objects on a network. Miller (1994), Okabe and Kitamura (1996), and Okunuki and Okabe (1998) formulate the Huff model on a network. Okabe *et al.* (1995) formulate the nearest neighbor distance method on a network. Miller (1999) proposes the space-time accessibility measure defined on a network. We elaborate those computational methods and apply them to the K-function method on a network.

The paper consists of five sections including this introductory section. In the next section, i.e. Section 2, we formulate the K-function method on a network. This method is used for testing the null hypothesis that points are uniformly distributed over a network, or testing no spatial interaction among the points. In Section 3 we formulate the cross K-function method on a network. This method is used for testing the null hypothesis that points of one kind are uniformly distributed regardless of the locations of points of another kind. For example, this method is useful to examine whether or not the distribution of fast-food stands is affected by the locations of subway stations. In Section 4, we develop a computational method for the network K-functions. We close this paper in Section 5, summarizing the major results.

2 The network K-function method

First let us fix a general setting. We consider a finite connected planar network, $\mathcal{N} = \mathcal{N}(L, Q)$, consisting of a set of links, $L = \{L_1, \dots, L_{n_L}\}$, and a set of nodes, $Q = \{q_1, \dots, q_{n_Q}\}$. We denote the whole links by L_T (i.e. $L_T = \bigcup_{k=1}^{n_L} L_k$) and the total length of L_T by l_T. Note that both ends of a link in L are nodes in Q and no links are connected except at nodes of Q. In the real world, the network $\mathcal{N}$ may represent a network of streets in a city where a node in Q represents an intersection of streets and a link in L represents a street segment between two intersections. We measure the distance between two points on L_T of $\mathcal{N}$ by the length of the shortest path between them, which we call the *network distance*. Note that we may refer to a network distance simply as a distance when a network is understood.

We next consider a set of n objects, such as fast-food stands, which can be regarded as a set of points, $P = \{p_1, \ldots, p_n\}$, and assume that the points of P are on the links L_T of $\mathcal{N}$. This implies that the size of the objects is negligibly small relative to the size of the network, and the objects are located along streets (imagine fast-food stands facing streets).

Now we assume that points of P are probabilistically distributed according to the uniform distribution over L_T of $\mathcal{N}$, i.e.,

$$f(p) = \begin{cases} \dfrac{1}{l_T}\ , & p \in L_T, \\ 0\ , & p \notin L_T, \end{cases} \tag{1}$$

where p is a point of P. Under this assumption, the probability of a point in P being placed on a subset, L_S, of L_T is proportional to the length, l_S, of L_S. Thus, the probability, $\Pr[N(L_S) = n_S]$, that the number, $N(L_S)$, of points of P being placed on L_S is exactly n_S is given by

$$\Pr\left[N(L_S) = n_S\right] = {}_nC_{n_S} \left(\frac{l_S}{l_T}\right)^{n_S} \left(1 - \frac{l_S}{l_T}\right)^{n-n_S}, \tag{2}$$

where ${}_nC_{n_S} = n!/\left(n_S!\,(n-n_S)!\right)$. Since this is a binomial distribution, the stochastic point process of P where points of P are distributed according to the uniform distribution of equation (1) is called the *binomial point process* (Ripley 1981).

The assumption of the binomial point process implies the hypothesis that objects represented by P(say fast-food stands) are uniformly and independently distributed over the street network $\mathcal{N}$. Thus if this hypothesis is not accepted, we may infer that the fast-food stands are spatially interacting and dependent each other; as a result, they form a non-uniform pattern. The stands may tend to be close together, or they may tend to keep away from each other. In the former case, we say that the fast-food stands are *clustering*; in the latter case, we say that they are *repelling*.

Under the assumption of the binomial point process, we define a function, $K(t)$, by

$$K(t) = \frac{1}{\omega}\,\mathrm{E}\left(\begin{array}{l}\text{the number of points of } P \text{ within} \\ \text{network distance } t \text{ of a point } p_i \text{ of } P\end{array}\right), \tag{3}$$

where $\mathrm{E}(\cdot)$ is the expected value with respect to $i = 1, \ldots, n$ ($p_i \in P$) where P follows the binomial point process, and ω is the density of points of P, i.e. $\omega = n/l_T$. We call this function $K(t)$ the *network K-function* for the binomial point process.

Since the points of P are distributed according to the binomial point process, all points in P are independently and uniformly distributed over L_T, and hence the above definition is equivalent to

$$K(t) = \frac{1}{\omega}\,\mathrm{E}\left(\begin{array}{l}\text{the number of points of } P \text{ within} \\ \text{network distance } t \text{ of a point } p \text{ on } L_T\end{array}\right), \tag{4}$$

where $\mathrm{E}(\cdot)$ is the expected value with respect to all possible locations of p over L_T. Let $L_p(t)$ be the subset of L_T where the network distance between p and any point on $L_p(t)$ is less than or equal to t; $l_p(t)$ be the length of $L_p(t)$. Since $\omega\, l_p(t)$ indicates the expected number of points of P on $L_p(t)$ under the assumption of the binomial point process, we can write $K(t)$ as

$$\begin{aligned} K(t) &= \frac{1}{\omega} \cdot \frac{1}{l_T} \int_{p \in L_T} \omega\; l_p(t)\,\mathrm{d}p \\ &= \frac{1}{l_T} \int_{p \in L_T} l_p(t)\,\mathrm{d}p\ , \end{aligned} \tag{5}$$

where dp implies the integration of p along L_T.

Now suppose that we observe n points of P on L_T (consequently, P is fixed), and let $L_{p_i}(t)$ be a subset of L_T where the network distance between p_i and any point on $L_{p_i}(t)$ is less than or equal to t. From equation (3), the K-function for this observed data set, $\hat{K}(t)$, is written as

$$\begin{aligned}\hat{K}(t) &= \frac{1}{\omega}\cdot\frac{1}{n}\sum_{i=1}^{n}(\text{the number of points of } P \text{ within network distance } t \text{ of } p_i) \\ &= \frac{1}{\omega}\cdot\frac{1}{n}\sum_{i=1}^{n}(\text{the number of points of } P \text{ on } L_{p_i}(t)). \qquad (6)\end{aligned}$$

Since $\omega = n/l_T$, we can rewrite $\hat{K}(t)$ as

$$\hat{K}(t) = \frac{l_T}{n^2}\sum_{i=1}^{n}(\text{the number of points of } P \text{ on } L_{p_i}(t)). \qquad (7)$$

We call the function $\hat{K}(t)$ the *observed network K-function* for a given set P.

Once we obtain the observed network K-function $\hat{K}(t)$ and the network K-function for the binomial point process $K(t)$, we can examine whether or not the observed points are distributed according to the binomial point process. If $\hat{K}(t) > K(t)$, we may infer that the points of P are clustering; if $\hat{K}(t) < K(t)$, the points of P are repelling.

We make one remark on the boundary effect. In general, we should distinguish two types of the boundary effect when we use a spatial statistic dealing with the distribution of points in a finite area, R. The first type boundary effect occurs when points outside R are neglected. The second type boundary effect occurs when a statistic which assumes a point process on an infinite area (such as the Poisson point process) is applied to a finite area R. The second type boundary effect occurs even when points are not distributed outside R. Any spatial statistic cannot correct the first type boundary effect unless points outside R are given. The second type boundary effect may be or may not be corrected. For example, consider the ordinary

K-function method defined by

$$K(t) = \frac{1}{\omega}\,\mathrm{E}\left(\begin{array}{l}\text{the number of points of } P \text{ within Euclidean} \\ \text{distance } t \text{ of } p_i \text{ of } P \text{ on an infinite area}\end{array}\right). \tag{8}$$

Under the assumption that the points of P are distributed according to the binomial point process, $K(t)$ is given by

$$\begin{aligned} K(t) &= \frac{1}{\omega}\,\mathrm{E}\left(\begin{array}{l}\text{the number of points of } P \text{ within} \\ \text{Euclidean distance } t \text{ of } p \text{ on an infinite area}\end{array}\right) \\ &= \frac{1}{\omega}(\omega\pi t^2) = \pi t^2. \end{aligned} \tag{9}$$

If this K-function method is applied to a finite area R, the second type boundary effect occurs, because the circle with radius t may not be included in R. To correct this boundary effect, Bailey and Gatrell (1995) adopt an adjustment factor which converts the finite area R to an infinite area. This modification is possible because the ordinary K-function method assumes an infinite homogeneous area. In the case of the network K-function method, however, we cannot adopt such an adjustment factor, because, first, a network is not homogeneous; second, it is difficult to consider an infinite homogeneous network. To overcome these difficulties, we need a method that directly deals with a finite inhomogeneous network. In the network K-function defined above, we consider this method in terms of $l_p(t)$ in theory. If this function $l_p(t)$ is practically computable, the proposed network K-function method would be useful in practice. We discuss this computational implementation in Section 4.

3 The network cross K-function method

In the preceding section, we dealt with one kind of points P. In this section we deal with two kinds of points, $P_a = \{p_{a1}, \ldots, p_{an_a}\}$ and $P_b = \{p_{b1}, \ldots, p_{bn_b}\}$, placed on L_T of $\mathcal{N}$. For instance, P_a may be a set of fast-food stands and P_b may be a set of subway stations. We are concerned with whether or not the locations of subway stations affect the distribution of fast-food stands. To examine this effect, we make a null hypothesis that points P_a are distributed according to the binomial point process. This assumption implies that fast-food stands P_a are uniformly distributed over L_T regardless the locations of subway stations P_b. Note that we make no assumption on the distribution of points P_b. If the above hypothesis is rejected, we may infer that the locations of subway stations P_b affect the distribution of fast-food stands P_a.

Extending the network K-function defined in Section 2, we define a new function, $K^{ba}(t)$, for two kinds of points P_a and P_b by

$$K^{ba}(t) = \frac{1}{\omega_a} \, \mathrm{E} \left(\begin{array}{c} \text{the number of points of } P_a \text{ within} \\ \text{network distance } t \text{ of a point } p_{bi} \text{ of } P_b \end{array} \right), \tag{10}$$

where $\mathrm{E}(\cdot)$ is the expected value with respect to $i = 1, \ldots, n_b$ $(p_{bi} \in P_b)$ where P_b follows the binomial point process, and ω_a is the density of points of P_a, i.e. $\omega_a = n_a/l_T$. We call this function $K^{ba}(t)$ the *network cross K-function of P_a relative to P_b* for the binomial point process.

Let $L_{p_{bi}}(t)$ be the subset of L_T where the distance between p_{bi} and any point on $L_{p_{bi}}(t)$ is less than or equal to t, and $l_{p_{bi}}(t)$ be the length of $L_{p_{bi}}(t)$. Under the assumption of the binomial point process, $\omega_a \, l_{p_{bi}}(t)$ gives the expected number of points of P_a on $L_{p_{bi}}(t)$. Thus we can rewrite $K^{ba}(t)$ as

$$\begin{aligned} K^{ba}(t) &= \frac{1}{\omega_a} \cdot \frac{1}{n_b} \sum_{i=1}^{n_b} \omega_a \, l_{p_{bi}}(t) \\ &= \frac{1}{n_b} \sum_{i=1}^{n_b} l_{p_{bi}}(t). \end{aligned} \tag{11}$$

Suppose that we observe the locations of n_a points of P_a and that of n_b points of P_b on L_T. For these observed data sets, we write the cross K-function corresponding to equation (10) as

$$\hat{K}^{ba}(t) \;=\; \frac{1}{\omega_a} \cdot \frac{1}{n_b} \sum_{i=1}^{n_b} (\text{the number of points of } P_a \text{ on } L_{p_{bi}}(t))\,. \qquad (12)$$

Since $\omega_a = n_a/l_T$, we can rewrite $\hat{K}^{ba}(t)$ as

$$\hat{K}^{ba}(t) \;=\; \frac{l_T}{n_a n_b} \sum_{i=1}^{n_b} (\text{the number of points of } P_a \text{ on } L_{p_{bi}}(t))\,. \qquad (13)$$

We call the function $\hat{K}^{ba}(t)$ the *observed network cross K-function of* P_a *relative to* P_b.

Once we obtain the observed network cross K-function $\hat{K}^{ba}(t)$ and the network cross K-function for the binomial point process $K^{ba}(t)$, we can examine whether or not the observed points P_a are distributed according to the binomial point process. If $\hat{K}^{ba}(t) > K^{ba}(t)$, we can infer that the points of P_a are clustering around the points of P_b; if $\hat{K}^{ba}(t) < K^{ba}(t)$, the points of P_a are repelling the points of P_b.

4 Computational methods for the network K-functions

In the preceding sections we theoretically obtained the equations of the network K-function and the network cross K-function, but those equations are implicit for actual computation. In this section, we wish to develop their computational methods by extending the methods proposed by Okabe *et al.* (1995), Okabe and Kitamura (1996), and Okunuki and Okabe (1998). Since a computational method for the network K-function is an extension of that for the network cross K-function, we first show the latter method and next the former method.

4.1 A computational method for the network cross K-function

We notice from equation (11) that $K^{ba}(t)$ is computable if the length $l_{p_{bi}}(t)$ of $L_{p_{bi}}(t)$ is computable for all points in $P_b = \{p_{b1}, \ldots, p_{bn_b}\}$. In this subsection, we develop a procedure for computing $l_{p_{bi}}(t)$ (illustrative examples in Figure 1 would be helpful to understand this procedure).

In Figure 1(a), $L_{p_{bi}}(t)$ (the subset of L where the distance from p_{bi} to any point on $L_{p_{bi}(t)}$ is less than or equal to t) is indicated by the thick gray-colored lines. Recalling that a network distance is the length of the shortest path between points, we notice that a set of the gray lines $L_{p_{bi}}(t)$ forms part of the 'extended' shortest path tree rooted at p_{bi}, and $l_{p_{bi}}(t)$ is given by the length of this gray tree. This observation suggests a strategy for computing $l_{p_{bi}}(t)$: first, we construct the 'extended' shortest path tree; second, we obtain $L_{p_{bi}}(t)$; third, we measure the length of $L_{p_{bi}}(t)$, which gives $l_{p_{bi}}(t)$.

To obtain the 'extended' shortest path tree, we first insert a point p_{bi} of P_b in the network $\mathcal{N}$ and regard it as a node (the black circle in Figure 1(b)). Second, we construct the shortest path tree rooted at p_{bi} (the thick black lines in Figure 1(b)). On each link that is not included in this shortest path tree (the thin lines in Figure 1 (b)), there exists one point to which two shortest paths from p_{bi} through the two end points of the link have the same distance (the stars in Figure 1(b)). We call this point a *break point* generated by p_{bi}. Third, we cut links at all break points, and put nodes on both cut ends (the white circles in Figure 1(c)). We call the resulting nodes *break-point nodes* generated by p_{bi}. Let $\mathcal{N}^{+i} = \mathcal{N}^{+i}(L^{+i}, Q^{+i})$ be the network modified through the above procedure, where L^{+i} denotes a set of resulting refined links, and Q^{+i} denotes a set of nodes consisting of Q and the resulting break-point nodes. Since the distance from the root p_{bi} to every node in Q^{+i} is known by the extended shortest path tree, we can easily obtain $L_{p_{bi}}(t)$.

We next measure the length $l_{p_{bi}}(t)$ of $L_{p_{bi}}(t)$ (the gray lines in Figure 1(d)). To this end, we treat $L_{p_{bi}}(t)$ link by link; we measure the length of $L_{p_{bi}}(t)$ for each link, and sum up those lengths. To be explicit, let $g_j^{p_{bi}}(t)$ be the length of $L_j \cap L_{p_{bi}}(t)$, i.e. the length of part of a link L_j that is included in $L_{p_{bi}}(t)$. Then we can obtain $l_{p_{bi}}(t)$ from

$$l_{p_{bi}}(t) = \sum_{j=1}^{n_L^{+i}} g_j^{p_{bi}}(t)\,, \tag{14}$$

where n_L^{+i} is the number of links in L^{+i}.

The form of $g_j^{p_{bi}}(t)$ varies according to the relation between L_j and $L_{p_{bi}}(t)$. Let l_j be the length of L_j, and q_{cj} and q_{rj} be the end points of L_j where $d(p_{bi}, q_{cj}) < d(p_{bi}, q_{rj})$ (for simplicity, we denote $d(p_{bi}, q_{cj})$ by d_{cj}, and $d(p_{bi}, q_{rj})$ by d_{rj}). When L_j is not included in $L_j(t)$, i.e. $L_j \cap L_{p_{bi}}(t) = \emptyset$, it is obvious that $g_j^{p_{bi}}(t) = 0$ (for example, L_1 in Figure 1(d)). When L_j is completely included in $L_{p_{bi}}(t)$, i.e. $L_j \cap L_{p_{bi}}(t) = L_j$, it is plain to see that $g_j^{p_{bi}}(t) = l_j$ (L_2 in Figure 1(d)). When L_j is partly included in $L_{p_{bi}}(t)$, $g_j^{p_{bi}}(t)$ is given by the length of $L_j \cap L_{p_{bi}}(t)$, i.e. $t - d_{cj}$ (L_3 in Figure 1(d)). In terms of these equations, $g_j^{p_{bi}}(t)$ is written as

$$g_j^{p_{bi}}(t) = \begin{cases} 0\,, & 0 \le t < d_{cj}\,, \\ t - d_{cj}\,, & d_{cj} \le t < d_{rj}\,, \\ d_{rj} - d_{cj} \quad (= l_j)\,, & d_{rj} \le t < \infty\,. \end{cases} \tag{15}$$

Therefore $K^{ba}(t)$ is explicitly written as

$$K^{ba}(t) \;=\; \frac{1}{n_b} \sum_{i=1}^{n_b} \sum_{j=1}^{n_L^{+i}} g_j^{p_{bi}}(t)\,. \tag{16}$$

Having established the computational procedure, let us consider the computational order of this procedure. Almost all of the computations in the above procedure have a linear order with respect to the number of links, but the computation of the shortest path tree has more order than that. Fredman and Tarjan (1984) show that the computational order of constructing the shortest path tree rooted at one node of a network is $O(n_Q \log n_Q)$, where n_Q is the number of nodes of the network. In the above procedure, we construct the shortest path tree rooted at each point of $P_b = \{p_{b1}, \ldots, p_{bn_b}\}$, and hence the total computational order is $O(n_b n_Q \log n_Q)$ (note that the number of nodes in Q^{+i} has the same order of n_Q). In usual applications, the number of points of P_b (say, subway stations) is much smaller than that of Q (the nodes of the network), and so the computational order is dominated by $O(n_Q \log n_Q)$.

4.2 A computational method for the network K-function

We notice from equation (5) that $K(t)$ is computable if the length $l_p(t)$ of $L_p(t)$ is computable for all possible locations of p over L_T of $\mathcal{N}$. In theory, a computational method for $l_p(t)$ is similar to that for $l_{p_{bi}}(t)$ of the network cross K-function shown in the above subsection, but the former method becomes much more complicated in practical computation. The reason is that in the cross K-function method, we compute $l_{p_{bi}}(t)$ for a finite number of points in P_b (i.e. $p_{b1}, \ldots, p_{bn_b}$), but in the K-function method, we have to compute $l_p(t)$ for every point p on L_T (infinite). We carry out this computation by moving p continuously along all links in L_T.

In Figure 2, the cross mark indicates the location of a point p on $\mathcal{N}$, and the thick gray lines indicate $L_p(t)$. As p moves along links, the form of $L_p(t)$ changes (compare Figure 2(a), (b), and (c)), but we notice that the topological relation of nodes in $L_p(t)$ does not change while p moves along a certain range (for example, Figure 2(a) and (b) have the same topological relation of nodes in $L_p(t)$; the topology remains the same while p moves between q_1 and q_2). Since this topological invariance property makes computation tractable, we first decompose L_T into refined links in such a way that the topology of $L_p(t)$ remains the same when p moves along each of the refined links. Once such refined links are obtained, we can obtain the length, $l_p(t|s)$, of $L_p(t)$ as a function of a parameter value, s, indicating the distance of p from one end point of the refined link on which p moves. Averaging the resulting function $l_p(t|s)$ with respect to s and all refined links, we obtain the expected length of $L_p(t)$ for every point p in L_T. This procedure consists of the following four steps (illustrative examples in Figure 3, 4, and 5 would be helpful to understand these steps). Note that in the following, the network $\mathcal{N}$ is assume to have no parallel links (but relaxation of this assumption is simple).

Step 1: refinement of links

First, we insert the break-points, Q^*, generated by all nodes of $Q = \{q_1, \ldots, q_{n_Q}\}$ in L (the white circles in Figure 3(b)). We call the resulting refined links the *first order refined links* and denote them by $L^* = \{L_1^*, \ldots, L_{n_L^*}^*\}$, where n_L^* is the number of links in L^*.

Now we choose an arbitrary first order refined link, L_k^*, in L^*, and suppose that p moves along this link from one end point, q_{kA}, to the other end point, q_{kB} (in Figure 3(c), the link $\overline{q_1^* q_2^*}$ is chosen as L_k^*). Further, we refine the first order refined links by inserting the break-point nodes, Q^{*k}, generated by q_{kA} and q_{kB} (the gray circles in Figure 3(c)). We call the resulting refined links the *second order refined links* with respect to L_k^* and denote them by $L^{*k} = \{L_1^{*k}, \ldots, L_{n_L^{*k}}^{*k}\}$, where n_L^{*k} is the number of links in L^{*k}. In general, indices of links in L^* and those of links in L^{*k} are different, but for notational convenience, we use the same index k for the link on which p moves (i.e. L_k^* and L_k^{*k} indicate the same link).

Step 2: classification of the second order refined links

Having obtained the second order refined links L^{*k}, we now classify them into four types.

Let l_j^{*k} be the length of the link L_j^{*k} ($j = 1, \ldots, n_L^{*k}$), and s be the network distance between p and q_{kA} ($0 \leq s \leq l_k^{*k}$). As p moves along L_k^{*k} from q_{kA} to q_{kB} (i.e. s varies from 0 to l_k^{*k}), each break point generated by p moves along a link in

L^{*k}. We call the traces of these break points the *trace links* of p. Note that each trace link is one of the second order refined links, and that the length of each trace link is equal to l_k^{*k}. In an illustrative example shown in Figure 3(d), as p moves from q_1^* to q_2^*, the break points, $p'_{(1)}$ and $p'_{(2)}$, generated by p move from q_1^{*k} to q_4 and from q_3 to q_2^{*k}, respectively. Thus, $\overline{q_1^{*k}q_4}$ and $\overline{q_3q_2^{*k}}$ are trace links of p.

Now we classify the second order refined links into four types (see Figure 4): Type 1 is L_k^{*k}, on which p moves, and Type 2 is the trace links of p. If we separate links of Type 1 and Type 2 from the network, the remaining network has two trees rooted at q_{kA} and q_{kB}, respectively. Type 3 links are the links forming the tree rooted at q_{kA}, and Type 4 links are those forming the tree rooted at q_{kB}.

Step 3: formulation of the length of $L_j^{*k} \cap L_p(t)$ for each type of links

We consider the length, $g_j^k(t|s)$, of $L_j^{*k} \cap L_p(t)$, i.e. the length of part of a link L_j^{*k} where the distance between p and any point on that part is less than or equal to t for each type of links. Illustrative examples are shown in Figure 5, where the cross mark indicates the location of p, the thick gray lines indicate $L_p(t)$, and the thick black lines on $L_p(t)$ indicate $L_j^{*k} \cap L_p(t)$. Note that q_{cj} and q_{rj} are the end points of L_j^{*k} where $d(q_{kA}, q_{cj}) < d(q_{kA}, q_{rj})$, and d_{cj} and d_{rj} indicate $d(q_{kA}, q_{cj})$ and $d(q_{kA}, q_{rj})$, respectively.

Type 1: L_k^{*k} (i.e. the link on which p moves)

Suppose that p is located on the left half of L_k^{*k} i.e. $0 \leq s \leq l_k^{*k}/2$. If $0 \leq t < s$, then $L_p(t)$ is completely included in L_k^{*k}, and so $g_k^k(t|s) = 2t$. If $s \leq t < (l_k^{*k} - s)$ (see Figure 5(a)), the right half of $L_p(t)$ is completely included in L_k^{*k}, but the left half of $L_p(t)$ is beyond q_{kA}. Thus $g_k^k(t|s) = t + s$. If $(l_k^{*k} - s) \leq t < \infty$, then L_k^{*k} is completely included in $L_p(t)$, and so $g_k^k(t|s) = l_k^{*k}$. In the same way, we can formulate $g_k^k(t|s)$ when p is located on the right half of L_k^{*k}. Summing up the above, we can write $g_k^k(t|s)$ as follows:

$0 \le s \le \frac{l_k^{*k}}{2}$

$$g_k^k(t|s) = \begin{cases} 2t \ , & 0 \le t < s \ , \\ t + s \ , & s \le t < l_k^{*k} - s \ , \\ l_k^{*k} \ , & l_k^{*k} - s \le t < \infty \ , \end{cases} \tag{17}$$

$\frac{l_k^{*k}}{2} < s \le l_k^{*k}$

$$g_k^k(t|s) = \begin{cases} 2t \ , & 0 \le t < l_k^{*k} - s \ , \\ t + (l_k^{*k} - s) \ , & l_k^{*k} - s \le t < s \ , \\ l_k^{*k} \ , & s \le t < \infty \ . \end{cases} \tag{18}$$

Type 2: the trace links of p (see Figure 5(b))

In this case, a shortest path from p to a point on a trace link may go through either q_{kA} or q_{kB} according to the value of s. Thus we can write $g_j^k(t|s)$ for links of Type 2 in the same way as in Type 1.

$0 \le s \le \frac{l_k^{*k}}{2}$

$$g_j^k(t|s) = \begin{cases} 0 \ , & 0 \le t < d_{cj} + s \ , \\ t - (d_{cj} + s) \ , & d_{cj} + s \le t < l_k^{*k} + d_{cj} - s \ , \\ 2t - (2d_{cj} + l_k^{*k}) \ , & l_k^{*k} + d_{cj} - s \le t < l_k^{*k} - d_{cj} \ , \\ l_k^{*k} \ , & l_k^{*k} - d_{cj} \le t < \infty \ , \end{cases} \tag{19}$$

$\frac{l_k^{*k}}{2} < s \le l_k^{*k}$

$$g_j^k(t|s) = \begin{cases} 0 \ , & 0 \le t < d_{cj} + l_k^{*k} - s \ , \\ t - (d_{cj} + l_k^{*k} - s) \ , & d_{cj} + l_k^{*k} - s \le t < d_{cj} + s \ , \\ 2t - (2d_{cj} + l_k^{*k}) \ , & d_{cj} + s \le t < l_k^{*k} - d_{cj} \\ l_k^{*k} \ , & l_k^{*k} - d_{cj} \le t < \infty \ . \end{cases} \tag{20}$$

In a similar fashion, we can formulate $g_j^k(t|s)$ for links of Type 3 and Type 4 as follows.

Type 3: the links included in the tree rooted at q_{kA} (see Figure 5(c))

$$g_j^k(t|s) = \begin{cases} 0 \ , & 0 \le t < d_{cj} + s \ , \\ t - (d_{cj} + s) \ , & d_{cj} + s \le t < d_{rj} + s \ , \\ d_{rj} - d_{cj} \quad (= l_j^{*k}) \ , & d_{rj} + s \le t < \infty \ . \end{cases} \tag{21}$$

Type 4: the links included in the tree rooted at q_{kB} (see Figure 5(d))

$$g_j^k(t|s) = \begin{cases} 0 \ , & 0 \le t < d_{cj} - s \ , \\ t - (d_{cj} - s) \ , & d_{cj} - s \le t < d_{rj} - s \ , \\ d_{rj} - d_{cj} \quad (= l_j^{*k}) \ , & d_{rj} - s \le t < \infty \ . \end{cases} \tag{22}$$

Step 4: integration of $g_j^k(t|s)$

As we showed above, we can formulate $g_j^k(t|s)$ for each second order refined link L_j^{*k} with respect to an arbitrary location of p on L_k^{*k}. Thus we can obtain the value, $l_p^k(t|s)$, of $l_p(t)$ when p is located at s on L_k^{*k} as

$$l_p^k(t|s) = \sum_{j=1}^{n_L^{*k}} g_j^k(t|s) \ . \tag{23}$$

Therefore, the expected value, $\mathrm{E}(l_p(t))$, of $l_p(t)$ for all possible locations of p over L_T is given by

$$\mathrm{E}(l_p(t)) = \frac{1}{l_T} \sum_{k=1}^{n_L^*} \int_0^{l_k^*} l_p^k(t|s) \, \mathrm{d}s \ , \tag{24}$$

and $K(t)$ in equation (5) can be rewritten as

$$\begin{aligned} K(t) &= \frac{1}{l_T} \int_{p \in L_T} l_p(t) \, \mathrm{d}p \ , \\ &= \frac{1}{l_T} \sum_{k=1}^{n_L^*} \int_0^{l_k^{*k}} l_p^k(t|s) \, \mathrm{d}s \ . \end{aligned} \tag{25}$$

We end up this subsection with the computational order of the above procedure. In the above procedure, we have to construct the shortest path tree rooted at each node of $Q = \{q_1, \ldots, q_{n_Q}\}$ and Q^* (note that the number of nodes in Q^* has at most the same order of n_Q). This computational order is $O(n_Q^2 \log n_Q)$. Just like the computational procedure in the preceding subsection, other computations in the above procedure have a linear order with respect to the number of links, so that the total computational order of the above procedure is dominated by $O(n_Q^2 \log n_Q)$.

5 Conclusions

The ordinary K-function method, which is often used in spatial statistics, assumes a homogeneous plane with the Euclidean distance. This assumption is hardly acceptable when we analyze the distribution of points in a fairly small district where the Euclidean distance is quite different from the shortest path distance on a street network and points are distributed along streets. To overcome this difficulty, this paper proposed the network K-funciton method and the network corss K-funciton method.

The network K-function method, defined by equations (5) and (7), is designed for testing the hypothesis that points are uniformly and independently distributed over a network (for instance, this statistic is useful for testing whether or not fast-food stands are uniformly distributed over streets in a downtown).

The network cross K-function method, defined by equations (11) and (13), is designed for testing the hypothesis that points of one kind are independently distributed on a network regardless of the locations of points of another kind (for instance, this statistics is useful for testing whether or not fast-food stands tend to gather around bus stops in a downtown).

These methods become practical by the computational methods shown in Section 4. The computational order of the network K-function method and that of the network cross K-function method are $O(n_Q^2 \log n_Q)$ and $O(n_Q \log n_Q)$, respectively, where n_Q is the number of nodes of the network.

We expect that these K-function methods provide a powerful tool for spatial analysis in a detailed geographical space with a street network, in particular, for micro-marketing.

Reference

Bailey, T.C., and Gatrell, A.C. (1995) *Interactive Spatial Data Analysis*, Harlow: Longman.

Buxton, T. (1992) GIS Expected to Micromarketing Challenge, *GIS World*, Vol.5, No.1, pp.70-72.

Cressie, N.A. (1993) *Statistics for Spatial Data*, Chichester: John Wiley.

Fotheringham, S., and Rogerson, P. (Editor)(1994) *Spatial Analysis and GIS (Technical Issues in Geographic Information Systems)*, London: Taylor & Francis.

Fredman, M.L., and Tarjan, R.E. (1984) Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Proceedings of the 25th Annual IEEE Symposium on Foundation of Computer Science*, Singer Island, Florida, pp.338-346.

Miller, H.J. (1993) GIS and the Geometry of Facility Location Problems, *GIS/LIS'93 Proceedings*, pp.530-539.

Miller, H.J. (1994) Market Area Delimination within Networks Using Geographic Information Systems, *Geographical Systems*, Vol.1, No.2, pp.157-173.

Miller, H.J. (1999) Measuring space-time accessibility benefits within transportation networks: Basic theory and computational methods, *Geographical Analysis*, Vol.31, No. , pp.187-212.

Okabe, A., and Fujii, A. (1984) The Statistical Analysis through a Computational Method of a Distribution of Points in Relation to Its Surrounding Network, *Environment and Planning A*, Vol.16, pp.107-114.

Okabe, A., Yomono, H. and Kitamura, M. (1995) Statistical Analysis of the Distribution of Points on a Network, *Geographical Analysis*, Vol.27, No. 2, pp.152-175.

Okabe, A., and Kitamura, M. (1996) A Computational Method for Market Area Analysis on a Network, *Geographical Analysis*, Vol.28, No.4, pp.330-349.

Okunuki, K., and Okabe, A. (1998) A computational method for optimizing the location of a store on a continuum of a network when users' choice behavior follows the Huff model, *Discussion Paper*, Center for Spatial Information Science at the University of Tokyo, No.19.

Ripley, D. (1981) *Spatial Statistics*, Chichester, John Wiley.

Upton, G., and Fingleton, B. (1985) *Spatial Data Analysis by Example: Volume 1: Point Pattern and Quantitative Data*, New York, John Wiley

Yomono, H. (1993) A Computational Method for Market Area Analysis on a Network, *Theory and Applications of GIS*, Vol.1, pp.47-56.

Figure captions

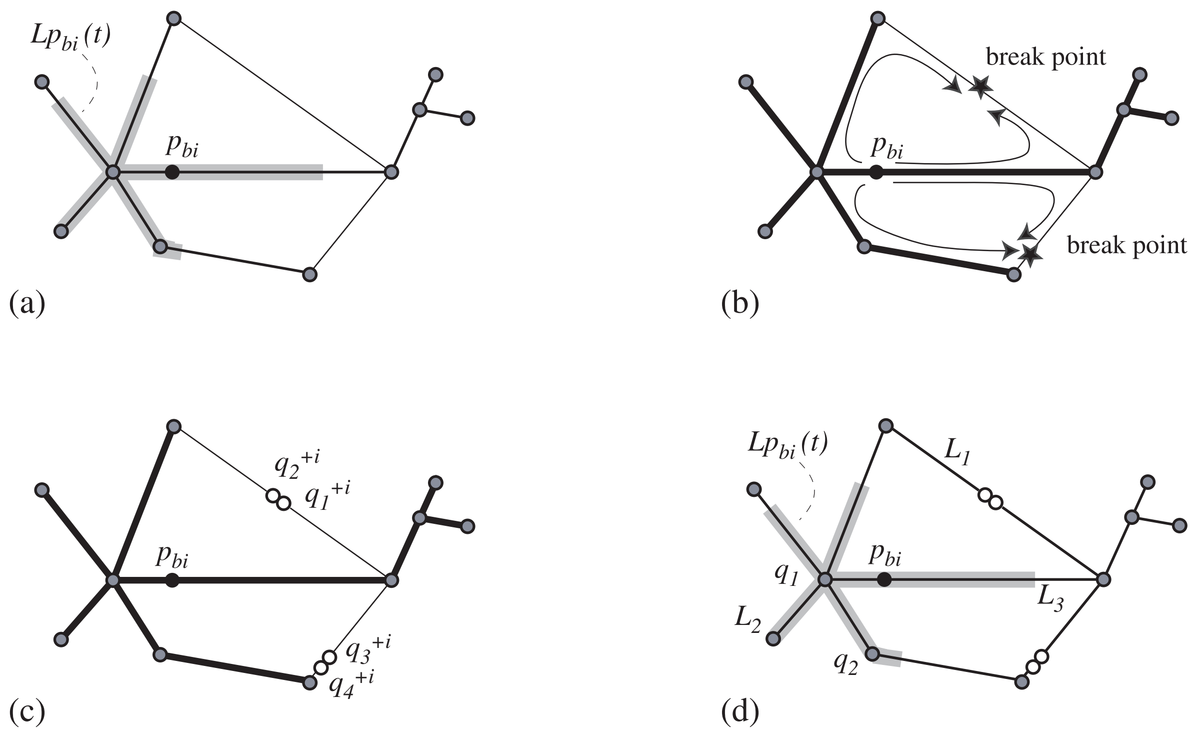

Figure 1: A computational method for the network cross *K*-function

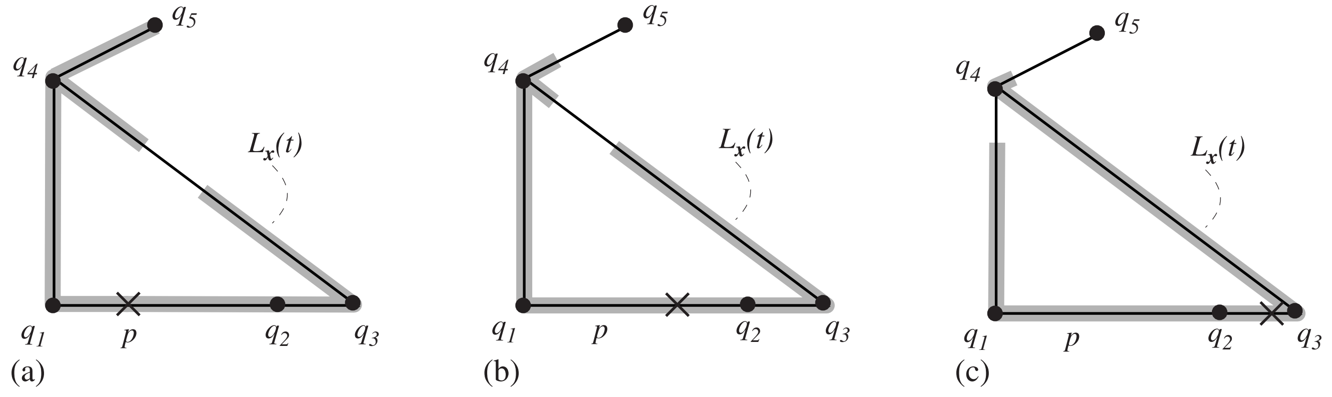

Figure 2: Topological relation of nodes

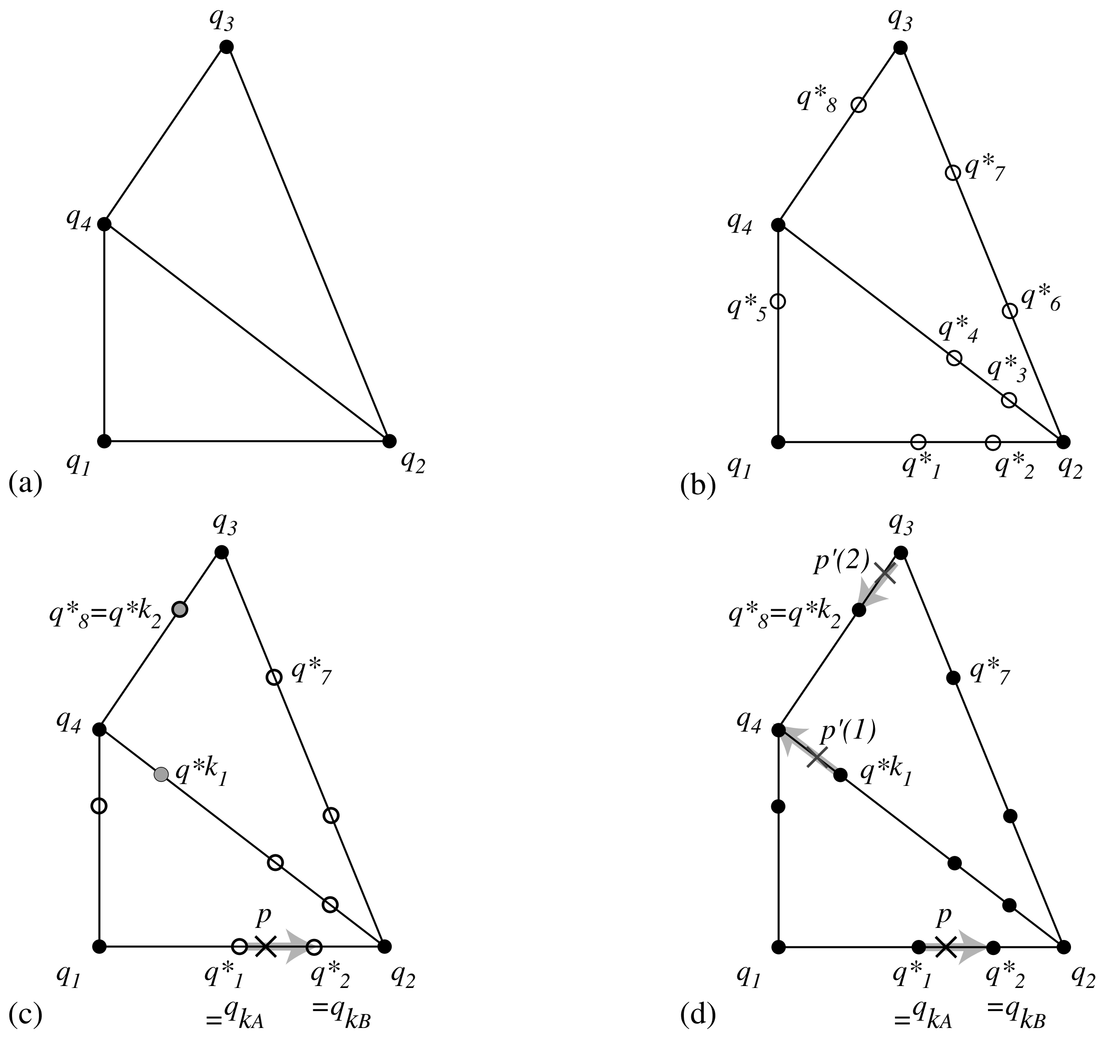

Figure 3: Refinement of links

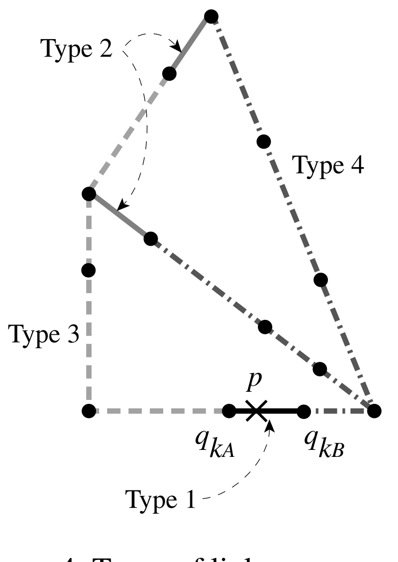

Figure 4: Types of links

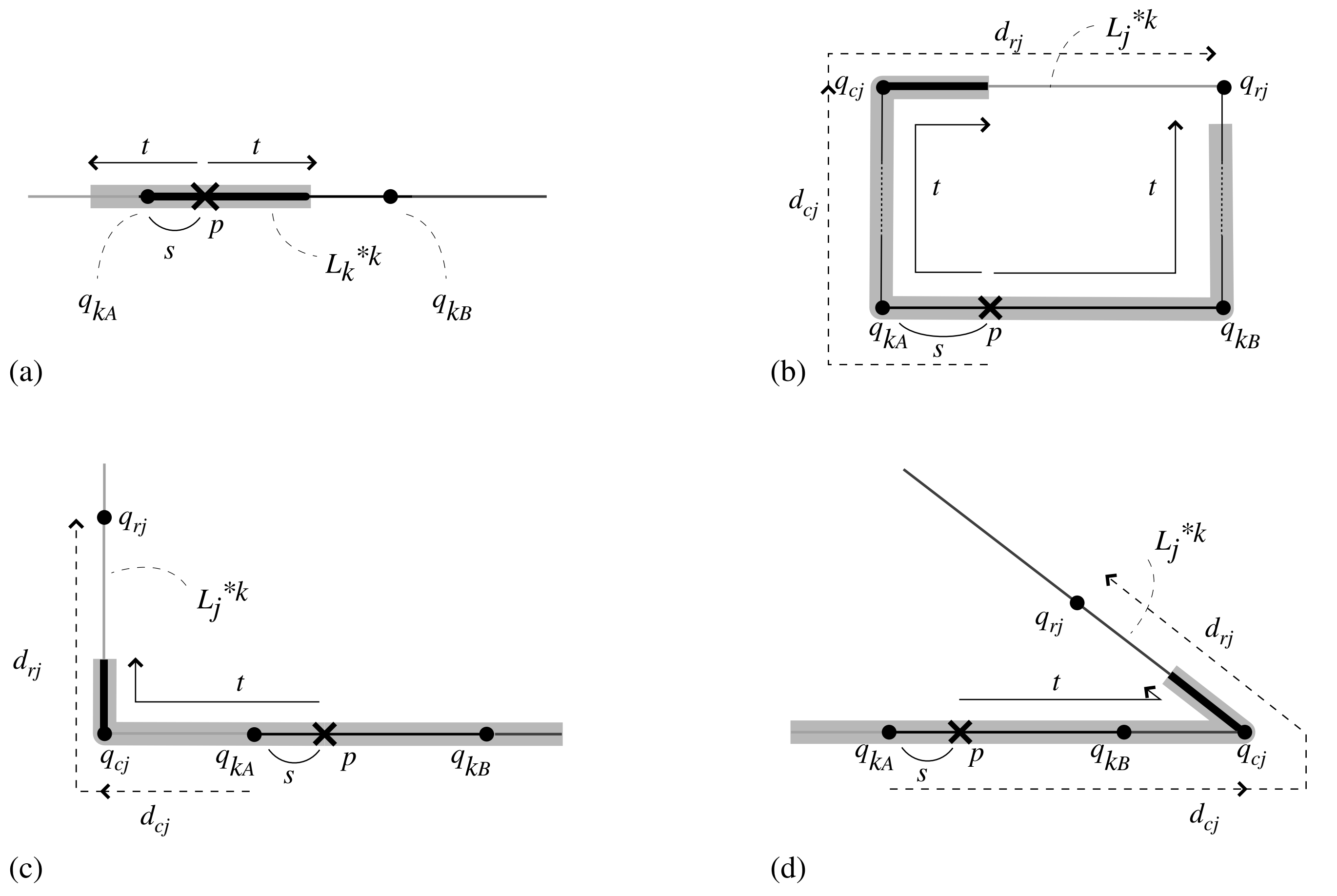

Figure 5: Computation of $g_j^k(t|s)$