A Voronoi Heuristic Approach to Dividing Networks into Equal-Sized Sub-Networks

Takehiro Furuta[1*], Atsuo Suzuki[2] and Atsuyuki Okabe[3]

[1]*Faculty of Engineering, Tokyo University of Science, 1-14-6 Kudankita, Chiyoda-ku, Tokyo 102-0073, Japan*
[2]*Faculty of Mathematical Sciences and Information Engineering, Nanzan University*
[3]*Faculty of Engineering, The University of Tokyo*
[*]*E-mail address: takef@fw.ipsj.or.jp*

(Received May 31, 2008; Accepted October 27, 2008)

We analyze the division of a connected network into p connected sub-networks with equal sizes in terms of network edge length. To find divisions, we minimize the sum of the absolute values of the difference between the average total edge length of p sub-networks from the total edge length of the network and the total edge length of each sub-network. Here, we propose a heuristic approach to the problem using a network Voronoi diagram and a linear programming formulation, and report computational experiments for actual road networks in Japan.
Key words: Network Division Problem, Network Voronoi Diagram, Linear Programming

1. Introduction

In this paper, we consider the problem of dividing a network into equal-sized sub-networks, where we assume that the network and the sub-networks are connected, and that network sizes are measured by the sum of their edge lengths. We assume that each edge represents not only the relation between the end vertices of the edge but also the infinite collection of points along it. It is important to consider this assumption in urban research, where many objects exist along network edges. For example, facilities exist on the edges of a road network. There have been numerous applications of this problem thus far (Shiode and Okabe, 2004; Suzuki and Drezner, 2008), including the following two examples. The first application is related to the equitable load model. Equitability is desirable for public and emergency systems. Baron *et al*. (2007) and Suzuki and Drezner (2008) proposed the equitable load model for an area. They divide an area into equal-sized sub-areas. However, in many cases we should consider the model on a network. Many objects in urban areas exist adjacent to streets or move along them. For example, response areas of an ambulance system should be along road networks, where patients are adjacent to streets and ambulances travel along them. If we divide the road network into equal-sized sub-networks and apply them to be the ambulance response area, the division attains equitability.

The second application is related to network cell count methods (Shiode and Okabe, 2004). Cell count methods (Haggett, 1965; Diggle, 2003) are usually applied to an area. The area is divided into same sized and same shaped sub-areas called cells. For the spatial analysis, we count the number of specific points (e.g. retail stores) in the cells. From that data, statistical and quantitative results about the point pattern can be computed. However, as in the case of the equitable load model, in many cases cells should be on the road network. The network cell count method uses equal-sized sub-networks, called network cells, and counts the number of specific points on the edges of each sub-network. This method requires the division of the network into equal-sized sub-networks.

There is little existing research related to this problem. To our knowledge, only Shiode and Okabe (2004) have examined it. They proposed a method to obtain several equal-sized sub-networks from an inner part of a given network. They consider two types of cells: a "proper cell" which has a given total edge length and is used in the network cell counting method and an "improper cell" which is a peripheral cell not used for cell counting. Their approach is based on a greedy algorithm where one proper cell is extracted from the given network at a time. In the process, this approach leaves behind many improper cells in the form of small sub-networks that do not have a given total edge length. As a result, many improper cells exist, and the number of proper cells to be extracted cannot be determined beforehand. Such properties of their method are serious drawbacks to the network cell count method as compared to the spatial cell count method.

In this paper, we adopt a different approach to computing network divisions. Given a positive integer p, we calculate the average total edge length of p sub-networks from the total edge length of the network. Given this value, we minimize the sum of the absolute values of the differences between the average and the sum of the edge lengths of each sub-network. We expect that the objective function value should be equal to zero, meaning that the p sub-networks are equally divided.

We propose an algorithm to solve this problem. The algorithm has two stages: the first obtains a rough approximate solution, and the second completes the solution. We use a network Voronoi diagram in the first stage, and a linear programming method for the second stage. The network Voronoi diagram (Okabe *et al*., 2008) is a generalized

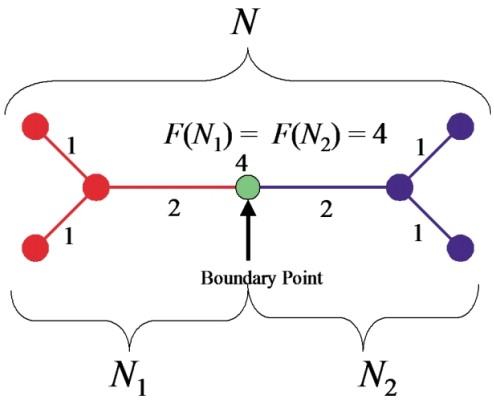

Fig. 1. Boundary point divides a connected network N into two equal-sized connected sub-networks N_1 and N_2.

version of the ordinary Voronoi diagram in the Euclidean plane (Okabe *et al.*, 2000). The network Voronoi diagram has a set of vertices called Voronoi generators that divide the network into sub-networks. Each sub-network is the set of points nearest to a given generator in the network, where nearness is measured as the shortest distance along the network. Linear programming (Hillier and Lieberman, 2005) is used to minimize a linear objective function of continuous real variables, subject to linear constraints.

This paper is organized as follows. In Sec. 2, we state the problem more formally and introduce the network Voronoi diagram that is used for our algorithm. In Sec. 3, we investigate the details of our algorithm. In Sec. 4, we show computational results using actual road network data. Finally, we give concluding remarks in Sec. 5.

2. Problem Statement

In this paper, we suppose the following network division problem. Given a connected network $N(V, E)$ with a set of vertices $V = \{v_1, \cdots, v_n\}$ and a set of edges E, our objective is to divide it into p equal-sized connected sub-networks, where p is a given positive integer. We use $F(N)$ as the total edge length of N. Using these values, we can calculate the average total edge length $F(N)/p$ of each sub-network. The objective of the problem is to find p equal-sized sub-networks N_k, that is,

$$F(N_k) = \frac{F(N)}{p}, N_k \cap N_j = \emptyset, k \neq j, k, j = 1, \cdots, p \tag{1}$$

where each sub-network is connected and has no points in common with another sub-network except at boundary points.

Figure 1 shows a trivial example of the problem. A connected network N is divided into two equal-sized connected sub-networks N_1 and N_2 by a boundary point. We have two remarks concerning this problem. First, the division that satisfies Eq. (1) is not necessarily unique, and in many cases there will be multiple solutions to the problem. Second, in some cases there will be no solutions, as in the case of Fig. 2.

We consider the problem of finding p equal-sized sub-networks as the following optimization problem:

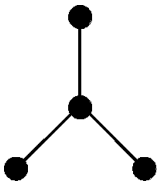

Fig. 2. Network has three edges with equal length. This network cannot be divided into two equal-sized connected sub-networks.

$$\text{minimize} \quad \sum_{k=1,\cdots,p} \left| \frac{F(N)}{p} - F(N_k) \right|, \tag{2}$$

$$\text{subject to} \quad N_k \cap N_j = \emptyset, \qquad k \neq j, k, j = 1, \cdots, p. \tag{3}$$

To solve the division problem, we need to retain the connectivity of each sub-network while having no intersection between them, except at boundary points. Under these conditions, our proposed algorithm uses a network Voronoi diagram (Okabe *et al.*, 2008) to find equal-sized connected sub-networks. Given a network and a set of generators on the network, the network Voronoi diagram defined by the generator set is a partition of the network. The partition consists of connected sub-networks called Voronoi sub-networks. Each point on a Voronoi sub-network is the closest to the generator point of the sub-network of all generator points, where closeness is determined as the shortest distance on the network, not on the Euclidean plane. In other words, each generator defines its associated Voronoi sub-network, which is a connected network surrounding it. We use the following notation to formally define the diagram:

$G = \{g_1, \cdots, g_p\}$: a set of generators ($G \subseteq V$),
$V(g_k)$: the Voronoi sub-network of generator g_k,
t : a point along an edge of N which may be a vertex of N,
$d(v_i, t)$: the shortest path distance on the network from v_i to t.

The definition of Voronoi sub-network $V(g_k)$ is as follows,

$$V(g_k) = \{t \mid d(g_k, t) \leq d(g_j, t), \forall j\}, k = 1, \cdots, p. \tag{4}$$

Then the set $\{V(g_1), \cdots, V(g_p)\}$ is the network Voronoi diagram. Using the diagram, the network is divided into Voronoi sub-networks. The boundary point b between two Voronoi sub-networks $V(g_k)$ and $V(g_j)$ satisfies the following condition,

$$d(g_k, b) = d(g_j, b). \tag{5}$$

Network Voronoi diagrams can be easily computed using the extended shortest path algorithm (Okabe *et al.*, 2008), which is based on Dijkstra's algorithm (Dijkstra, 1959). The shortest path tree is defined as the set of edges connecting all vertices such that the sum of the edge lengths from a given start vertex to each vertex is minimized. The tree spans all vertices, but does not cover all edges. To construct the network Voronoi diagram, all points on the uncovered edges must be assigned to their closest generators. Okabe

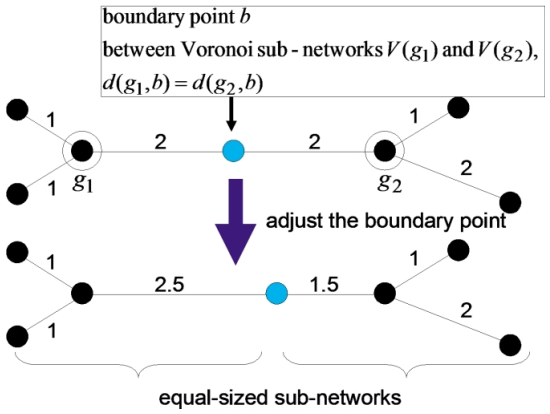

Fig. 3. Example of adjustment of boundary points. Boundary point is moved to achieve equal-sized division.

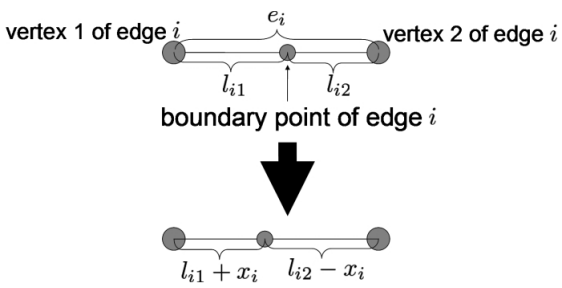

Fig. 4. Decision variable x_i decides the value of movement of boundary point of edge i.

et al. (2008) extend the tree to cover those edges that are not covered by the tree and the shortest path algorithm to compute the extended tree. An outline of the extended algorithm is as follows. First, we add a dummy vertex that connects all generators as the start vertex of the shortest path algorithm. Next, we compute the extended shortest path algorithm from the dummy vertex until all points on the network are assigned to a generator. The property of the shortest path algorithm that decides the shortest path in order of proximity ensures that each point is assigned to its closest generator (see Okabe *et al*. (2008) for details of this algorithm). In the next section, we propose an algorithm for computing the equal-sized division problem using the diagram.

3. Solution Algorithm

We developed a two-stage algorithm for computing the equal-sized sub-network mentioned above. In the first stage, to achieve an approximately equal-sized division, we consider the network Voronoi diagram that minimizes Eq. (2). To define the network Voronoi diagram, we need to obtain the p generators that generate such a diagram. The following local search algorithm is used for finding the generators.

[Voronoi Heuristics]

Step 1. *Initialization*

Randomly select p generators g_k, $k = 1, \cdots, p$ from among the vertices in N, and construct $V(g_k)$. Set $U = \infty$ to retain the best value of the objective.

Step 2. *Termination condition*

Repeat 3 until U does not decrease.

Step 3. *Search generators*

For each k, select a vertex g'_k in $V(g_k)$, and construct the network Voronoi diagram by $g_1, \cdots, g'_k, \cdots, g_p$. If g'_k gives the minimum of

$$u = \sum_{i=1,\cdots,p,i\neq k} \left| \frac{F(N)}{p} - F(V(g_i)) \right| + \left| \frac{F(N)}{p} - F(V(g'_k)) \right| \quad (6)$$

then set $g_k = g'_k$ and $U = u$.

Step 4. *Follow up*

In each k, $N_k = V_k$.

In Step 3 we search for a vertex in the set of vertices $V(g_k)$ that makes the largest improvement. The output of this algorithm is a local optimum. Then we adopt a multiple start method, that is, we start Voronoi heuristics with various randomly selected p generators and confirm that our solution is the best of them. The solution is not exact in many cases however, and so we consider the following finishing up method.

The network Voronoi diagram has many boundary points that represent the boundary between Voronoi sub-networks (Eq. (5)). We adjust the position of the boundary points to complete the solutions (Fig. 3). When boundary points are adjusted, they are moved only along their edge to create equal-sized divisions. We formulate the adjustment problem as a mathematical programming formulation. To describe the formulation, the following additional notations are used.

B : the index set of edges having a boundary point,
e_i : the length of edge $i \in B$,
l_{ij} : the length from a boundary point to the end vertex $j = 1, 2$ of edge i ($e_i = l_{i1} + l_{i2}$),
M_{ijk} : equals 1 if vertex j of edge i is assigned to sub-network N_k, otherwise 0,
L_k : the sum of the length of edges which have no boundary point in N_k.

Furthermore, we use decision variables x_i such that the boundary point of edge i moves x_i from its current position (Fig. 4). Then the adjustment problem can be formulated as follows:

$$\text{minimize} \sum_{k=1,\cdots,p} \left| \frac{F(N)}{p} - \left\{ L_k + \sum_{i\in B} M_{i1k}(l_{i1} + x_i) + \sum_{i\in B} M_{i2k}(l_{i2} - x_i) \right\} \right|, \quad (7)$$

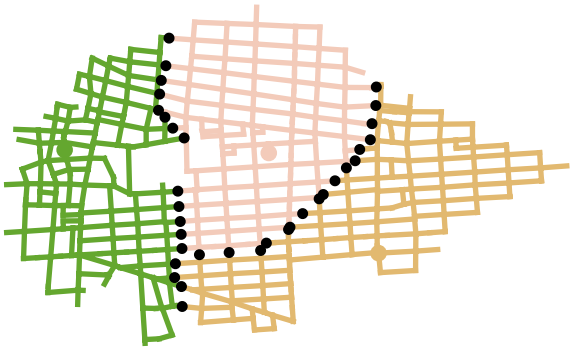

Fig. 5. Results of Voronoi heuristics. Large circles denote Voronoi generators and small circles denote boundary points between sub-networks. Number of vertices is 400 and $p = 3$.

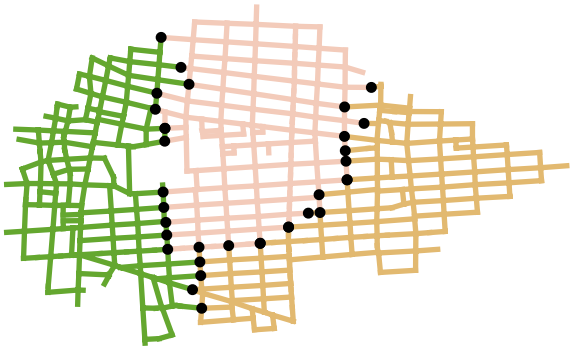

Fig. 6. Results of the linear programming method. Black circles denote boundary points between sub-networks.

$$\text{subject to} \quad 0 \le l_{i1} + x_i \le e_i, \qquad i \in B, \quad (8)$$

$$x_i \in \mathbb{R}, \qquad i \in B. \quad (9)$$

The objective function (7) is the sum of the absolute values of the difference between the average total edge length and the total edge length of each sub-network whose boundary points are moved by x_i. Constraints (8) ensure that the boundary point moves only along the edge.

This mathematical formulation is not an ordinary linear programming formulation. Generally, it is not easy to solve this kind formulation. However, this formulation can be easily transformed into a linear programming formulation using the additional decision variables z_k, such that

$$\left| \frac{F(N)}{p} - \left\{ L_k + \sum_{i \in B} M_{i1k}(l_{i1} + x_i) + \sum_{i \in B} M_{i2k}(l_{i2} - x_i) \right\} \right| \le z_k. \quad (10)$$

To minimize the sum of z_k is equivalent to minimizing the objective function. Therefore, we can transform the formulation to a linear programming formulation using z_k as follows:

minimize

$$\sum_{k=1,\cdots,p} z_k, \quad (11)$$

subject to

$$\frac{F(N)}{p} - \left\{ L_k + \sum_{i \in B} M_{i1k}(l_{i1} + x_i) + \sum_{i \in B} M_{i2k}(l_{i2} - x_i) \right\} \le z_k, \qquad k = 1, \cdots, p, \quad (12)$$

$$\frac{F(N)}{p} - \left\{ L_k + \sum_{i \in B} M_{i1k}(l_{i1} + x_i) + \sum_{i \in B} M_{i2k}(l_{i2} - x_i) \right\} \ge -z_k, \qquad k = 1, \cdots, p, \quad (13)$$

$$z_k \ge 0, z_k \in \mathbb{R}, \qquad k = 1, \cdots, p, \quad (14)$$

in addition to (8) and (9).

We use the Voronoi heuristics with multiple starts to get an approximate solution, and then the linear programming formulation to improve the solution.

4. Computational Experiments

Programs were coded in Java using the optimization software package ILOG CPLEX 10.0 to solve the problem which was formulated as a linear programming problem. The programs were run on a Pentium 4, 2.53 GHz with 512 MB RAM. We made ten different network types for our experiments using the actual road network of Nagoya City in

Table 1. Details for 3 types of networks (500, 700, and 900 vertices).

id	n	p	$F(N)/p$	vor obj.	vor time (s)	lp obj.	lp time (s)
1	500	5	1093.16	28.12	54.67	0.00	0.34
2	500	5	1147.31	39.39	59.30	0.00	0.36
3	500	5	1078.64	25.29	52.91	0.00	0.34
4	500	5	1096.20	29.45	52.03	0.00	0.36
5	500	5	1117.82	21.80	57.27	0.00	0.34
1	500	15	364.39	93.32	77.41	0.00	0.38
2	500	15	382.44	111.88	80.55	0.00	0.39
3	500	15	359.55	122.90	76.86	0.00	0.38
4	500	15	365.40	99.84	76.81	0.00	0.38
5	500	15	372.61	85.08	75.02	0.00	0.38
1	500	25	218.63	137.34	88.08	0.00	0.41
2	500	25	229.46	187.31	94.02	0.00	0.42
3	500	25	215.73	125.95	94.00	0.00	0.41
4	500	25	219.24	146.53	90.30	0.00	0.42
5	500	25	223.56	248.62	97.69	34.19	0.39
1	700	5	1555.46	25.92	111.75	0.00	0.38
2	700	5	1593.01	37.46	114.08	0.00	0.38
3	700	5	1515.82	52.55	110.70	0.00	0.38
4	700	5	1497.68	30.48	109.30	0.00	0.39
5	700	5	1543.12	23.99	114.33	0.00	0.36
1	700	15	518.49	137.08	149.70	0.00	0.41
2	700	15	531.00	101.90	158.92	0.00	0.41
3	700	15	505.27	121.65	160.39	0.00	0.41
4	700	15	499.23	148.44	149.31	0.00	0.41
5	700	15	514.37	147.17	151.09	0.00	0.41
1	700	25	311.09	175.47	181.47	0.00	0.41
2	700	25	318.60	169.49	179.34	0.00	0.45
3	700	25	303.16	238.59	177.61	0.00	0.42
4	700	25	299.54	224.43	169.56	0.00	0.42
5	700	25	308.62	186.34	184.42	0.00	0.44
1	900	5	2033.46	28.79	203.58	0.00	0.36
2	900	5	2103.14	31.76	192.20	0.00	0.34
3	900	5	2070.65	20.37	192.53	0.00	0.34
4	900	5	2111.33	48.37	209.63	0.00	0.38
5	900	5	2061.26	47.74	188.94	0.00	0.36
1	900	15	677.82	187.11	305.09	0.00	0.39
2	900	15	701.05	147.25	274.31	0.00	0.39
3	900	15	690.22	171.51	268.78	25.22	0.39
4	900	15	703.78	177.36	278.89	22.24	0.39
5	900	15	687.09	107.12	272.61	0.00	0.41
1	900	25	406.69	263.66	339.34	90.46	0.45
2	900	25	420.63	253.22	327.80	0.00	0.44
3	900	25	414.13	322.99	326.50	22.24	0.44
4	900	25	422.27	283.24	331.86	6.99	0.44
5	900	25	412.25	332.62	334.28	15.95	0.47

Aichi prefecture. We select a vertex of the original network randomly. Then we extract it with its incident edges and neighboring vertices and construct the sample network. The number of vertices of each sample network varies from 100 to 1000. We made 5 examples for each type. Figures 5 and 6 show an example of our computational results for a network having 400 vertices with $p = 3$. Figure 5 shows the computational results of the Voronoi heuristics, where the large circles denote the Voronoi generators when the approximate solution is found. Figure 6 shows the results of the linear programming method using the results shown in Fig. 5. Boundary points are moved to achieve the equal-sized sub-network. In this case, the network is divided into equal-sized sub-networks exactly.

Table 1 shows experimental results for networks with 500, 700, and 900 vertices. The column labeled "id" shows

Table 2. Summary of experimental results for 100 to 500 vertices.

		Voronoi Heuristics				Linear Programming				
		% over the ave.				% over the ave.				
n	p	max-min	max-max	sd	time (s)	max-min	max-max	sd	time (s)	# exact
100	5	0.56	3.05	2.84	2.24	0.00	0.00	0.00	0.35	5
100	10	0.66	7.83	3.37	2.95	0.00	0.00	0.00	0.36	5
100	15	1.01	14.29	3.97	3.82	0.00	0.00	0.00	0.37	5
100	20	1.85	29.19	5.00	4.17	0.00	15.05	0.93	0.40	2
100	25	0.95	46.06	5.22	4.55	0.00	25.80	1.17	0.42	1
100	30	1.45	42.39	5.28	4.50	0.00	34.78	1.60	0.39	2
200	5	0.47	2.78	4.50	9.66	0.00	0.00	0.00	0.35	5
200	10	0.34	5.42	5.51	12.36	0.00	2.29	0.49	0.36	4
200	15	0.22	10.90	6.07	14.41	0.00	2.36	0.48	0.37	3
200	20	0.19	11.66	4.90	16.26	0.00	9.55	1.02	0.38	3
200	25	0.70	15.39	5.23	17.41	0.00	24.02	1.86	0.39	2
200	30	0.65	29.17	5.13	18.52	0.00	31.43	2.02	0.41	2
300	5	0.19	2.23	5.43	20.48	0.00	0.00	0.00	0.35	5
300	10	0.34	4.49	5.90	25.60	0.00	0.79	0.23	0.37	4
300	15	0.47	7.25	6.06	30.05	0.00	7.59	1.24	0.37	4
300	20	0.49	11.99	6.54	32.79	0.00	14.46	2.49	0.38	2
300	25	0.34	11.60	6.24	34.99	0.00	15.41	2.38	0.39	1
300	30	0.30	17.33	7.24	36.40	0.00	22.93	4.03	0.41	0
400	5	0.10	1.33	4.32	36.70	0.00	0.00	0.00	0.36	5
400	10	0.38	4.82	7.70	44.55	0.00	0.00	0.00	0.36	5
400	15	0.36	6.14	8.07	50.23	0.00	3.49	0.72	0.37	4
400	20	0.52	9.82	7.71	54.61	0.00	4.28	1.09	0.39	3
400	25	0.59	18.19	8.65	60.72	0.00	17.41	4.29	0.40	2
400	30	0.41	22.48	6.95	63.58	0.00	18.56	1.50	0.41	3
500	5	0.27	1.37	6.75	55.23	0.00	0.00	0.00	0.35	5
500	10	0.37	3.55	8.62	69.44	0.00	0.00	0.00	0.36	5
500	15	0.08	6.02	8.35	77.33	0.00	0.00	0.00	0.38	5
500	20	0.38	8.94	8.18	85.46	0.00	1.80	0.28	0.39	4
500	25	0.29	12.73	8.53	92.82	0.00	7.65	0.84	0.41	4
500	30	0.52	12.09	8.46	97.05	0.00	6.50	1.14	0.45	3

the identifier for each network. The column labeled "n" shows the number of vertices in the road network. The columns labeled "vor obj." and "vor time" show the value of the objective function at the end of Voronoi heuristics and the computational time, respectively. The columns labeled "lp obj." and "lp time" show the value of the objective function at the end of our linear programming method and the computational time, respectively. In the table, the numbers in the "lp obj." column are equal to 0. This means that the network is divided into p equal-sized sub-networks.

Tables 2 and 3 summarize our experimental results. The results of the Voronoi heuristics and the linear programming method are shown in the columns labeled "Voronoi Heuristics" and "Linear Programming", respectively. The column labeled "max-min" shows the maximum value of $\frac{\left\{\min_{k=1,\cdots,p}\left|\frac{F(N)}{p}-F(N_k)\right|\right\}}{\frac{F(N)}{p}} \times 100$ in five networks. This value represents the max-min percentage over the average edge length. The column labeled "max-max" shows the maximum value of $\frac{\left\{\max_{k=1,\cdots,p}\left|\frac{F(N)}{p}-F(N_k)\right|\right\}}{\frac{F(N)}{p}} \times 100$ in five networks. The averages of the standard deviations of the total edge length of sub-networks over the five networks are shown in the column labeled "sd". The column labeled "time (s)" shows the computational time. In our experiments, the number of starts of the Voronoi heuristics is 20 and the numbers in the column labeled "time (s)" in the column labeled "Voronoi heuristics" shows the total computational time for the multiple starts method. The column labeled "# exact" shows the number of road networks that are divided into exactly equal-sized sub-networks among the five road networks.

The tables show that the linear programming method improves the solution of Voronoi heuristics drastically. Many networks were divided into equal-sized sub-networks within a reasonable computational time, especially for small values of p. When the value of p is large, the results of the Voronoi heuristics are not good in some cases, and we cannot get exact solutions. However, the value of the standard deviation is very small. Our Voronoi heuristics are simple local search algorithms and the number of can-

Table 3. Summary of our experimental results for 600 to 1000 vertices.

		Voronoi Heuristics				Linear Programming				
		% over the ave.				% over the ave.				
n	p	max-min	max-max	sd	time (s)	max-min	max-max	sd	time (s)	# exact
600	5	0.20	1.08	7.46	81.34	0.00	0.00	0.00	0.36	5
600	10	0.41	3.84	9.35	103.37	0.00	0.00	0.00	0.36	5
600	15	0.11	8.24	10.61	117.05	0.00	6.68	2.14	0.38	4
600	20	0.68	7.14	10.02	133.11	0.00	1.54	0.29	0.43	4
600	25	0.88	9.04	9.82	134.38	0.00	3.74	0.86	0.44	3
600	30	0.45	14.77	9.42	139.65	0.00	14.73	3.20	0.43	1
700	5	0.12	0.80	7.92	112.03	0.00	0.00	0.00	0.38	5
700	10	0.24	2.45	10.56	139.71	0.00	0.00	0.00	0.39	5
700	15	0.16	3.71	10.48	153.88	0.00	0.00	0.00	0.41	5
700	20	0.13	5.94	10.24	167.69	0.00	0.00	0.00	0.42	5
700	25	0.19	7.21	9.72	178.48	0.00	0.00	0.00	0.43	5
700	30	0.17	8.76	9.95	188.29	0.00	1.28	1.05	0.43	3
800	5	0.12	1.24	7.48	149.27	0.00	0.00	0.00	0.37	5
800	10	0.41	3.49	11.99	180.35	0.00	0.00	0.00	0.39	5
800	15	0.48	6.02	12.55	212.84	0.00	0.00	0.00	0.38	5
800	20	0.21	5.92	11.82	229.22	0.00	0.27	0.08	0.40	4
800	25	0.33	11.81	12.54	240.43	0.00	7.25	2.44	0.44	3
800	30	0.15	11.16	10.63	264.86	0.00	6.23	0.87	0.44	4
900	5	0.10	0.82	7.99	197.38	0.00	0.00	0.00	0.36	5
900	10	0.20	3.84	13.71	252.34	0.00	0.00	0.00	0.38	5
900	15	0.07	6.20	13.13	279.94	0.00	1.83	1.61	0.39	3
900	20	0.56	6.69	13.34	305.72	0.00	0.00	0.00	0.40	5
900	25	0.71	9.69	14.17	331.96	0.00	8.07	2.91	0.45	1
900	30	0.50	11.93	12.38	340.15	0.00	0.65	0.15	0.44	3
1000	5	0.15	0.88	7.25	239.84	0.00	0.00	0.00	0.36	5
1000	10	0.28	2.68	12.22	319.42	0.00	0.00	0.00	3.18	5
1000	15	0.37	4.93	13.56	355.17	0.00	0.00	0.00	0.40	5
1000	20	0.41	7.67	13.70	387.08	0.00	1.66	0.53	1.81	4
1000	25	0.21	8.62	14.04	408.19	0.00	11.86	5.71	0.50	1
1000	30	0.47	18.22	14.68	425.99	0.00	15.76	5.73	0.44	1

didate vertices in each iteration is dependent on the number of vertices of Voronoi sub-networks. The average number of vertices is small in that case.

5. Conclusions

We proposed a new approach for the equal-sized network division problem. The problem is to find equal-sized connected sub-networks within a given connected network. First, the problem is transformed into an optimization problem, where the objective is to minimize the sum of the absolute values of the difference between the average size and the size of each sub-network. We designed a heuristic algorithm having two stages. In the first stage, the network Voronoi diagram is used to compute the approximate division of a network. We gave a linear programming formulation to improve the division. Our approach performed very well in computational experiments, where many networks were divided into equal-sized sub-networks exactly, and we obtained approximate solutions for the other networks.

References

Baron, O., Berman, O., Krass, D. and Wang, Q. (2007) The equitable location problem on the plane, *European Journal of Operational Research*, **183**(2), 578–590.

Diggle, P. J. (2003) *Statistical Analysis of Spatial Point Patterns*, 2nd Ed., Oxford University Press, New York.

Dijkstra, E. W. (1959) A note on two problems in connexion with graphs, *Numerische Mathematik*, **1**, 269–271.

Haggett, P. (1965) *Locational Analysis in Human Geography*, Edward Arnold, London.

Hillier, F. S. and Lieberman, G. J. (2005) *Introduction to Operations Research*, 8th Ed., McGraw-Hill, New York.

Okabe, A., Boots, B., Sugihara, K. and Chiu, S. N. (2000) *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd Ed., John Wiley & Sons, New York.

Okabe, A., Satoh, T., Furuta, T., Suzuki, A. and Okano, K. (2008) Generalized network voronoi diagrams: Concepts, computational methods and applications, *International Journal of Geographical Information Science*, **22**, 965–994.

Shiode, S. and Okabe, A. (2004) Analysis of point patterns using the network cell count method, *GIS—Theory and Applications—*, **12**, 57–66 (in Japanese).

Suzuki, A. and Drezner, Z. (2008) The minimum equitable radius location problem with continuous demand, *European Journal of Operational Research*, doi:10.1016/j.ejor.2008.01.022.