

NETWORK VORONOI DIAGRAMS

Margot Graf and Stephan Winter
Institut für Geoinformation, Technische Universität Wien
Gusshausstr. 27-29, 1040 Wien
Tel. +43 1 58801 12712, Fax +43 1 58801 12799
margot.graf@aon.at, winter@geoinfo.tuwien.ac.at

Abstract

This paper presents and implements the Network Voronoi diagram. Dijkstra's shortest path algorithm is modified in way that it calculates shortest paths from several Voronoi generators at the same time. The result is a partition of the nodes of the network. Additionally the arcs of the network are attributed to the generators, considering especially their direction and asymmetric costs. Faces are excluded in the diagrams because the considered networks are not planar. Three applications demonstrate the contribution of Network Voronoi diagrams compared to Voronoi diagrams.

1 EINLEITUNG

Imagine you are looking for a school for your kid. Among the criteria to be considered will be the length of the way to school. If you formulate this as a spatial analysis problem, you are looking for the school that is closest to your home, among all schools in your city. The classical approach to solve this spatial analysis problem is the Voronoi diagram. The Voronoi diagram isolates the area that is closest to each school; typically it is calculated on the map plane with the L_2 metric (Fig. 1). Your home will be in exactly one of these identified catchment areas.



Fig. 1: Three schools and their catchment areas in form of a Voronoi diagram.

Sometime the (popular) approach is of limited value, especially if the possibilities to move in space are limited to one or several networks. In this case, the above described method gives only rough estimates and might even be significantly wrong. Several assumptions of the Voronoi diagram are violated in urban areas:

- Distances between two addresses are not Euclidean; they have to be measured along the travel network(s). If your daughter has to walk around a block of buildings, the way to school can be significantly higher than the Euclidean distance.
- Distances can be asymmetric. Imagine your daughter walks downhill in one direction, but uphill on her way home. If she uses a bus, then the bus route in both directions can vary due to one-way streets, and thus the travel times will be different.
- Distances can be inhomogeneous. In principle, your daughter can explore the urban space walking. But if she uses for parts of her way to school the subway, she will travel over long distances in relatively short time. Isochrones isolate in this case areas that are no longer in any case connected.

Now, which school is really the closest, that means, which school can be reached in the shortest time, considering all the travel modes offered in urban areas that can be used by kids? This question is answered in the literature here and there with references to the Network Voronoi diagram. The Network Voronoi diagram considers distances (or, more general, travel costs) only in networks, not in the plane. It divides the network, not the space, into Voronoi cells. A Voronoi cell in a network is the set of nodes and edges that is closer to one Voronoi generator (here: a school) than to any other (Fig. 2).



Fig. 2: The Network Voronoi diagram for the same three generators as in Figure 1, calculated on walking times along the street network.

There are many relevant problems for Network Voronoi diagrams. In urban areas catchment areas are calculated frequently for, e.g., businesses, or infrastructure facilities. Applications can be classified into three groups:

- With a static Network Voronoi diagram questions for the next generator can be solved. Which school is next to my home? Or reverse: which ambulance is the closest to an accident? The first problem is of the kind traveling to the generator, the second traveling from the generator. In case of asymmetric distances in the network the Network Voronoi cells can be different in these cases.
- With the Network Voronoi diagram we can simulate consequences of changes in the network. The selection of a new location is simulated by introducing a new generator in the Network Voronoi diagram (or, if the facility will be closed, the case is simulated by deleting a generator). The position of the generator can be varied, and the changes in the Network Voronoi cell structure will allow assessing different location alternatives. For example, questions can be: How do catchment areas of schools change if a new school opens its doors? Or what is the influence of a construction site, blocking a street for months, on the catchment area of a specific facility?
- The Network Voronoi diagram can react dynamically on changes in the network, which is critical in real-time problems. If in a network nodes or edges are blocked, or

their travel costs significantly change, how do the catchment areas change? Imagine for example a fire in a building. If one emergency exit is blocked by the fire: to which alternative exits shall the people be guided from the different areas in the building?

In this paper the Network Voronoi diagram will be presented and implemented. For that purpose the shortest path algorithm of Dijkstra will be modified such that it calculates shortest paths from all generators in parallel. By this way we calculate forests of shortest path trees, which allows to partition the nodes of the network. Having a Node Network Voronoi diagram, we can assign the edges of the network to generators. For these Edge Network Voronoi diagrams we consider the directions of the edges, and asymmetric cost attributes. Other applications know also an Area Network Voronoi diagram, which is discounted here because the urban travel networks are typically non-planar and thus violate the fundamental condition for this type of Network Voronoi diagram. Three applications will demonstrate the usefulness of the derived diagrams. The algorithm and the flexible management of directed and asymmetric edges are the original contributions of this paper.

2 PREVIOUS WORK

The differences between the calculation of a (planar) Voronoi diagram and a Network Voronoi diagram will be presented. References to relevant literature will be given.

2.1 Voronoi Diagrams

Voronoi diagrams (Aurenhammer 1991; Aurenhammer and Klein 2000) are described frequently in the context of the so-called *Post Office Problem* (de Berg et al. 2000). On a plane there are some point-like facilities, let us say, post offices. For each post office an area shall be assigned that contains all points of the plane closer to the assigned post office than to any other. The post office functions a Voronoi generator, and the boundaries of the identified area represent the Voronoi diagram. The shape to the Voronoi diagram depends on the used metric for the defining condition. Some prominent distance measures for Voronoi diagrams are the *Euclidean distance* (L_2 metric) or the *Manhattan distance* (L_1 metric).

The classical and simplest form of a Voronoi diagram is explained with Figure 3. In this case the L_2 metric is applied in the plane. In the L_2 metric the distance between two points A and E is calculated according to the Formula 2.1:

$$L_2(A, E) = \sqrt{(x_A - x_E)^2 + (y_A - y_E)^2} \quad (2.1)$$

The choice of the L_2 metric makes the planar space isotropic: traveling in any direction is possible with the same speed. Points of the same distance from a generator form a circle. The size of the circle is dependent from (a) the traveling speed, and (b) the time spent traveling. For that reason the circle is called an isochrone: isochrones connect points of the same distance from one or many generating points. Drawing the isochrones of several generators for the same point in time creates points of the Voronoi diagram in the intersection of the isochrones (Fig. 3).

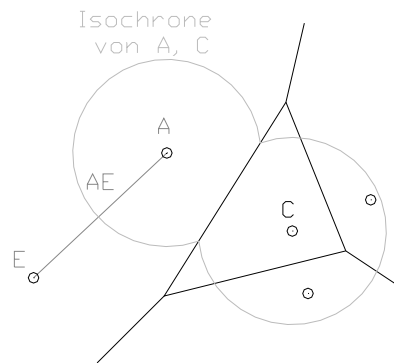


Fig. 3: The Euclidean distance AE , and the isochrones for the two Voronoi generators A and C : the Voronoi edge is generated by the intersection points of the isochrones over time.

This kind of calculation of the Voronoi diagram is the conventional implementation in Geographic Information Systems (GIS), although it does not represent adequately real travel costs in transport oriented problems. The more the transportation network is anisotropic, the less realistic is the approximation of travel costs by a Voronoi diagram. We will show in Section 4 the difference between a Voronoi diagram and a Network Voronoi diagram in a realistic travel network.

The partitioning of the plane by a Voronoi diagram is even less realistic in multi-modal transportation networks. Such networks are frequently not planar, and the different travel modes allow traveling with a broad range of speeds along the network edges. Furthermore the access to the network is limited: the street network can be entered only at house entrances, and the public transportation network can be entered only at stops or stations. For that reason the Network Voronoi diagram will be introduced.

2.2 Network Voronoi diagrams

A network is defined by its *nodes* and the weighted and directed connecting relations, the *edges*. Creating a Network Voronoi diagram, the focus is on the partitioning of the nodes and edges, not of the plane. In principle there are three types of Network Voronoi diagrams (Fig. 4, Okabe et al. 2000).

- The *Node Network Voronoi diagram*: It assigns each node in a network to a generator by the rule that the network distance from the node to the generator is smaller than to any other generator in the network.
- The *Edge Network Voronoi diagram*: It assigns each edge in a network to single generator, or it splits edges into two parts such that the points on one part of the edge are assigned to one generator (which is closest to these points), and the points on the other part of the edge are assigned to another generator (which is closer to them).
- The *Area Network Voronoi diagram*: the partitioning of the area is done in a way such that each point is assigned to its closest generator, considered the minimal Euclidean distance to the next network point, and from that point network distances. This diagram is different to the planar Voronoi diagram described above, and it requires a planar network.

to the neighbor are compared, and the node will be labeled with the smaller costs. The algorithm ends when all nodes are labeled finally. Each generator produces a shortest path tree. When labeling a node, the algorithm registers additionally from which neighbor node it was reached, with other words, to which shortest path tree it belongs. Hence we know the complete partition of the set of nodes. Consider the following example. Given a network with six nodes and eleven edges (Fig. 5), we are interested in the partition of the nodes into two sets, generated by nodes 1 and 5.

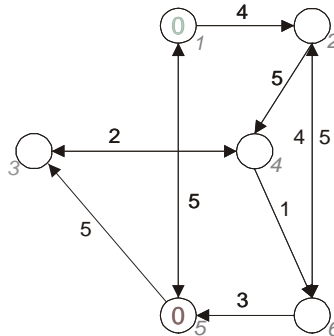


Fig. 5: Example of a simple network.

The network can be represented also in form of an adjacency matrix:

$$Adjazenzmatrix = \begin{bmatrix} \infty & 4 & \infty & \infty & 5 & \infty \\ \infty & \infty & \infty & 5 & \infty & 5 \\ \infty & \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 4 \\ 5 & \infty & 5 & \infty & \infty & \infty \\ \infty & 4 & \infty & \infty & 3 & \infty \end{bmatrix}$$

For the algorithm three vectors are initialized: (a) a vector *Distanz*, containing the currently known cheapest traveling costs from the closest generators, is initialized by 0 for all generators, and infinity for all other nodes; (b) a vector *Knoten*, which is binary and represents in Boolean form whether the label is temporary, is initialized with 1; and (c) a vector *KV*, which contains for each node its closest generator, is initialized with the node IDs for the generators, and infinity for all other nodes. For the network in Fig. 5 the three vectors are:

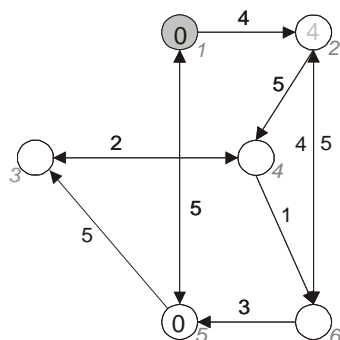
$$Distanz = [0 \quad \infty \quad \infty \quad \infty \quad 0 \quad \infty]$$

$$Knoten = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$$

$$KV = [1 \quad \infty \quad \infty \quad \infty \quad 5 \quad \infty]$$

The algorithm starts with these three vectors, and repeats the described iterations until the vector *Distanz* contains no other finite values. For the network in our example there are six iterations in total. In each iteration, a node *minindex* is selected that shows the minimal value in the vector *Distanz* and is not final, i.e., its value in *Knoten* is still 1. For this node, all non-final neighbors are determined from the adjacency matrix and the vector *Knoten*.

1. Iteration: *minindex* = 1, *j* = {2, 5}

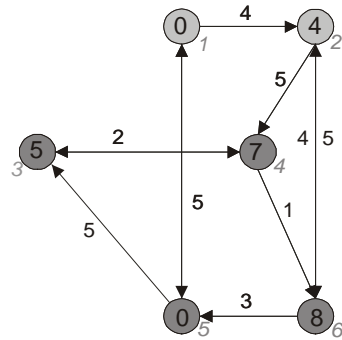


$$Distanz = [0 \quad 4 \quad \infty \quad \infty \quad 0 \quad \infty]$$

$$Knoten = [0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$$

$$KV = [1 \quad 1 \quad \infty \quad \infty \quad 5 \quad \infty]$$

After six iterations we receive the following result:



$$Distanz = [0 \ 4 \ 5 \ 7 \ 0 \ 8]$$

$$Knoten = [0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$KV = [1 \ 1 \ 5 \ 5 \ 5 \ 5]$$

After having determined the Node Network Voronoi diagram, we can start assigning edges to the generators.

3.2 Determination of the Edge Network Voronoi diagram

Each edge is labeled according to the Voronoi generators of its start and end node. To do so four different cases can occur and need separate treatment (Fig. 6):

1. Start and end node belong the same generator in the Node Network Voronoi diagram. In this case the edge is assigned to that shortest path tree, and labeled with the generator ID.
2. Start and end node belong to different generators, and the edge is one-directional. In this case one cannot travel from any point on the edge to the start node, or reverse, any point on the edge can be reached only from the start node. The whole edge has to be assigned to the shortest path tree of the start node.
3. Start and end node belong to different generators, the edge is bi-directional but symmetric, and the edge can be entered only at start or end node. Such cases occur for example in the public transportation network, where start and end node of an edge represent stops. In this case the whole edge is assigned to the shortest path tree of the start or end node, whatever is cheaper.
4. Start and end node belong to different generators, the edge is bi-directional and not necessarily symmetric, and entering the edge is possible at any point. In this case the edge has to be split. The split point is the point at which traveling costs to the generator of the start node are equal to the traveling costs to the generator of the end node. The first part x_A (start node to split point) will be assigned to the shortest path tree of the start node, and the second part, x_E , from the split point to the end node, will be assigned to the shortest path tree of the end node. At the splitting point a node will be introduced that is not assigned to any shortest path tree. It represents the boundary between two Network Voronoi cells.

Case 1	
Case 2	
Case 3	

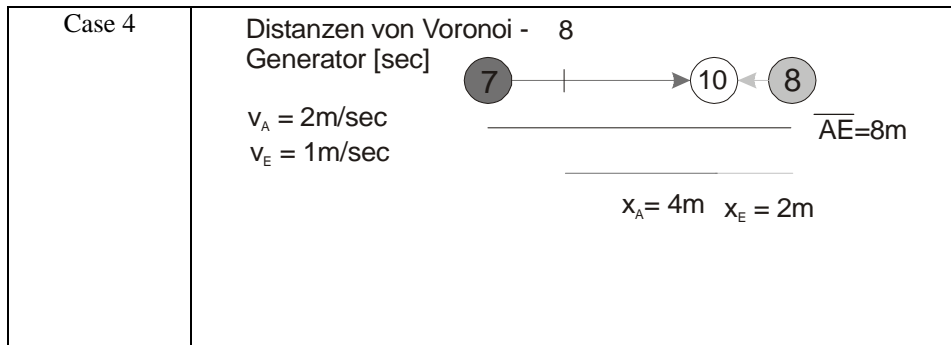


Abbildung 6: Die Einteilung einer Kante zu den Voronoi-Generatoren ihrer beiden Knoten.

Consider again the network in Fig. 5. The Node Network Voronoi diagram of its six nodes is already known and stored in the vector KV . The vector $Distanz$ contains the cheapest traveling costs from the generators to the nodes in the network. Another table, $Koordinaten$, might contain the coordinates of the nodes.

$$\begin{array}{l}
 \begin{matrix} \left[\begin{array}{cc} 2 & 10 \\ 5 & 10 \\ \dots & \dots \\ \dots & \dots \\ 2 & 0 \\ 5 & 0 \end{array} \right] \\
 \text{Koordinaten} = \end{matrix}
 \end{array}
 \quad
 \begin{array}{l}
 Distanz = [0 \ 4 \ 5 \ 7 \ 0 \ 8] \\
 KV = [1 \ 1 \ 5 \ 5 \ 5 \ 5]
 \end{array}$$

The network contains eleven edges. The start nodes are stored in a vector A , and the end nodes are stored in a vector E . A and E are constructed in the following way. The adjacency matrix is scanned row-by-row for finite values. For example, the first finite value is at the position $[1, 2]$, and thus, $A[1] = 1$, and $E[1] = 2$: the first found edge starts in node 1 and ends in node 2. Having vectors A and E , we can determine the vectors $KV[A]$ and $KV[E]$. They contain the generators of the start and end node, respectively.

$$\begin{array}{ll}
 k = 11 & KV[A] = [1 \ 1 \ 1 \ 1 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5] \\
 A = [1 \ 1 \ 2 \ 2 \ 3 \ 4 \ 4 \ 5 \ 5 \ 6 \ 6] & KV[E] = [1 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5 \ 1 \ 5 \ 1 \ 5] \\
 E = [2 \ 5 \ 4 \ 6 \ 4 \ 3 \ 6 \ 1 \ 3 \ 2 \ 5] & Kantenvoronoi = [\infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty]
 \end{array}$$

The determination of the assignment of the edges is done sequentially for $i = 1, 2, \dots, k$. Thus, the vector $Kantenvoronoi$ is filled from the first to the last position. The result will be:

- For each edge we get the corresponding generator in the vector $Kantenvoronoi$:
 $Kantenvoronoi = [1 \ 1 \ 1 \ 1 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5]$
- For edges that had to be split, we get the updated vector E :
 $E = [2 \ 7 \ 4 \ 8 \ 4 \ 3 \ 6 \ 7 \ 3 \ 8 \ 5]$
- For edges that had to be split, also the vector $Distanz$ is updated:
 $Distanz = [1 \ 0 \ 4 \ 5 \ 7 \ 2 \ 0 \ 8 \ 2.5 \ 8.5]$
- Finally, split nodes are registered in an updated vector $Koordinaten$:

$$\begin{array}{l}
 \begin{matrix} \left[\begin{array}{cc} 2 & 10 \\ 5 & 10 \\ \dots & \dots \\ \dots & \dots \\ 2 & 0 \\ 5 & 0 \\ \underline{2} & \underline{5} \\ \underline{5} & \underline{1.1} \end{array} \right] \\
 \text{Koordinaten} = \end{matrix}
 \end{array}$$

The result of the process for the example network is shown in Fig. 7.

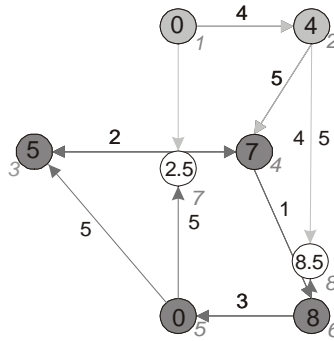


Fig. 7: The Edge Network Voronoi diagram of the network in Fig. 5.

4 IMPLEMENTATION AND TEST

The described algorithm for the determination of Node and Edge Network Voronoi diagram was implemented in IDL, the Interactive Data Language of Research Systems™. This language is especially useful for rapid prototyping and testing of algorithms that exploit the extensive library of matrix and image processing operations. In our case, the calculation of the plane Voronoi diagram, the isochrones, and the visualization of the network are taken from the library.

For tests of the implementation two data sets were used:

- A street network data set of the inner districts of Vienna. This network was extended by four subway lines (lines *U1* to *U4*).
- An indoor network of the university building Gusshausstrasse 27-29, created from existent 2D-CAD-data of the single floor plans. This data set was kindly provided by the Bundesimmobilienverwaltung.

Figure 8 shows the Network Voronoi diagram of four arbitrarily chosen generators in the street network data set. It is represented by four different colors in the network. Additionally the figure shows the plane Voronoi diagram of the area, represented by the thicker black lines. Finally the shading of the figure indicates in intervals of 5 minutes the shortest traveling times from the generators to the nodes (the darker the shading, the longer the travel). For the travel time calculation a mean speed of 20 km/h was assumed.

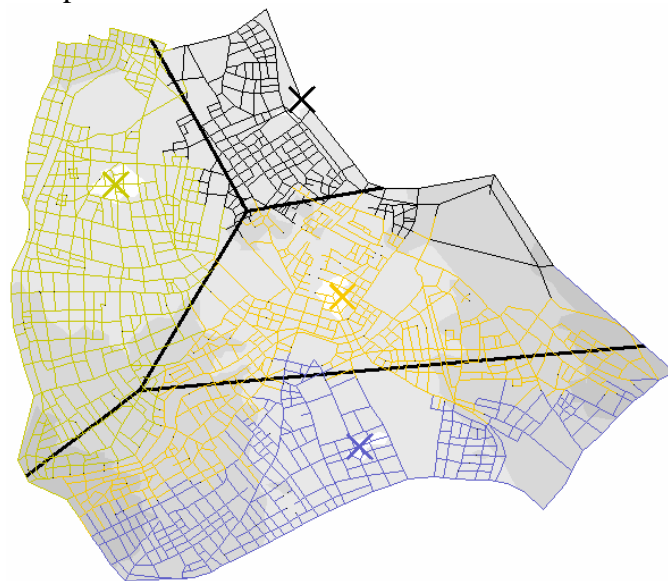


Fig. 8: Network Voronoi Diagram (colored network) and plane Voronoi Diagram (thick black lines) of four arbitrary generators.

The figure allows to demonstrate the difference between both types of diagrams. The difference is significant in areas that are developed by an inhomogeneous network. Consider the blue generator in Fig. 8: the generator is located in an area which is characterized by many one-way streets in its West, and a lesser dense network in its East. Both factors downgrade the reachability of the neighbored network parts. This shortens the catchment area of the blue generator for the benefit of the orange one.

For Fig. 9 we calculated the catchment areas of five high schools, assuming that the pupils travel by foot or using the subway. Again the shading represents travel times in 5 minute intervals. It turns out that the maximal distance of a point in the network to closest school is about 30 minutes. The average travel time is 14.4 minutes. Additionally one can calculate the size of the catchment areas, or from that other socio-demographic factors. For example the catchment area of the red generator consists of a network of 98 km street length. – Figure 9 shows still the plane Voronoi diagram, although the network is no longer planar, and highly anisotropic due to the different travel modes.

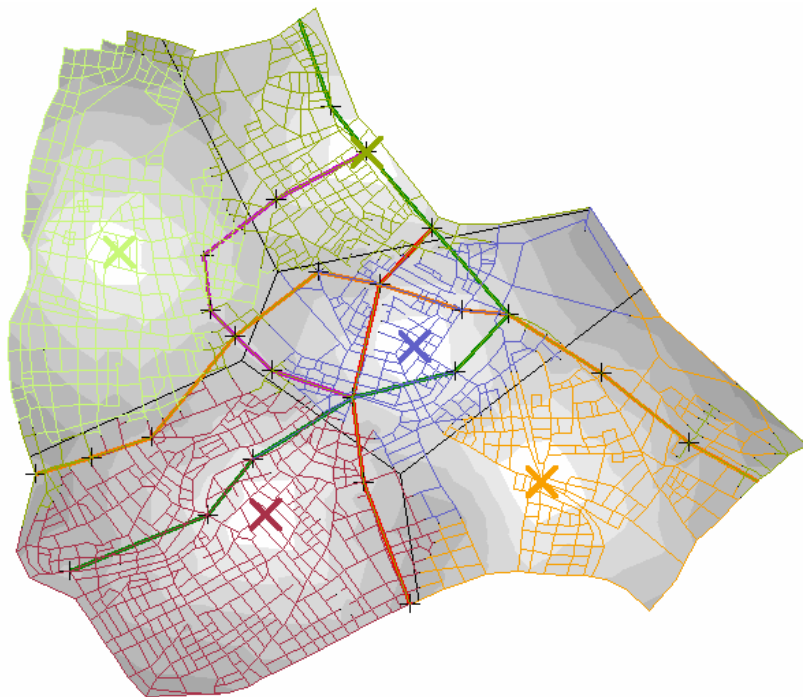


Fig. 9: Network Voronoi Diagram (colored) and Voronoi Diagram (thick gray lines) of five generators in a symmetric pedestrian network and an integrated subway network (thick colored lines).

In an indoor network the calculation of the Network Voronoi diagram becomes relevant for example in disaster events. Assume that a fire blocks the main entrance / exit of the building Gusshausstr. 27-29 (Fig. 10), and fire and smoke are spreading. The building has to be evacuated, but many signs in the floors of the building still point towards the (blocked) main exit. In fact the planners of the building partitioned the floors into a Network Voronoi diagram when planning for the emergency directions. The main exit has a defined catchment area. But now, in case of this fire, the available number of emergency exits (generators) has changed, and thus, the whole Network Voronoi diagram has changed dramatically. For people in the former catchment area of the main entrance the most important question is: which is the next emergency exit from my actual position now?

Figure 10 (left) shows the regular Network Voronoi diagram, and Fig. 10 (right) shows the actual one. It can easily be recognized that blocking the main entry enlarges mainly the catchment area of the black generator. In normal circumstances the black generator serves as an emergency exit for 308 m (15%) of the floors of the building. With a blocked main exit this number increases to 548 m (27%). – For these considerations we have neglected ways in

rooms, and connections between rooms. This indoor network is no longer planar, and no comparison with a plane Voronoi diagram can be made.

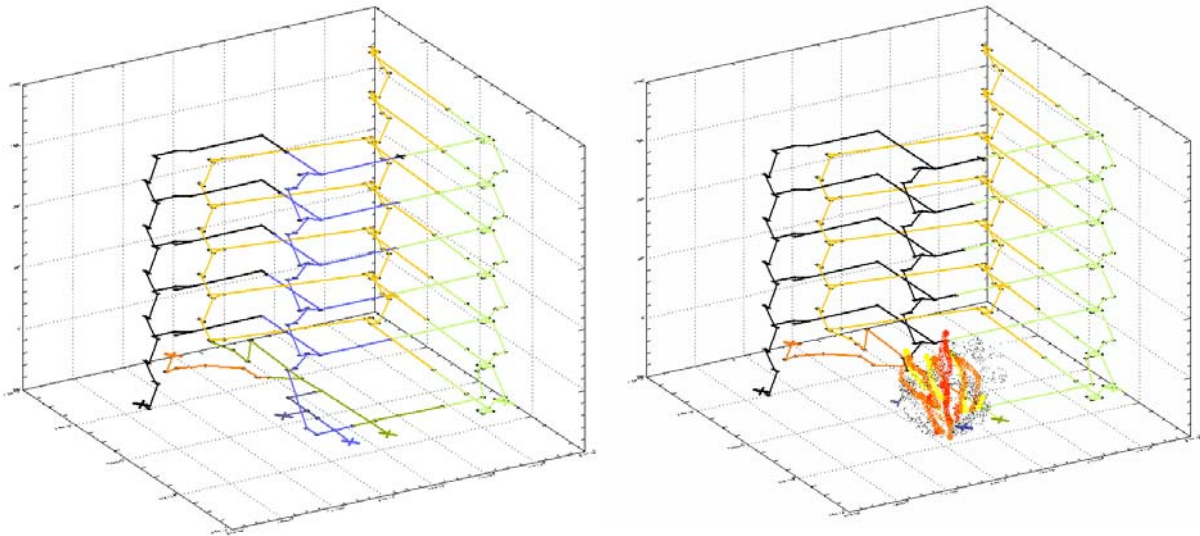


Fig. 10: Indoor Network Voronoi Diagram, with generators at all exits of the building (left). If an exit is blocked (right) the diagram changes dramatically.

5 DISCUSSION UND OUTLOOK

The principles of a plane Voronoi diagram was transferred to networks. The shortest path algorithm of Dijkstra was modified for this purpose. The modified version calculates shortest travel times from selected nodes – the Voronoi generators – to all other nodes in the network, and determines the closest generator for each node. For the points along the edges also the next generator could be determined. Hence, the Network Voronoi diagram is complete for all points in the network. The modification is presented in the paper, and implemented for tests. Applications follow mainly for inhomogeneous and for anisotropic networks (multi-modal traveling), for which we could demonstrate a significant increase of precision in the analysis compared to the plane Voronoi diagram. Other applications are in three dimensional networks, like indoor networks, which cannot be covered by plane Voronoi diagrams at all. We have presented examples for the applications that show the relevance and superiority of this approach.

The presented algorithm can be applied with any network. The implementation could be improved with respect to efficiency in storage or computation. More relevant seems an extension to consider generators located somewhere in the network, not necessarily on nodes. Another extension concerns the shortest routes. With the determination of the diagram one could store the shortest routes to the generators, not only the generator ID. Analysing the shortest routes afterwards could show network segments that are frequently used, which is especially useful in the simulation and planning of disaster events.

Additional factors can be included in the analysis of a Network Voronoi diagram, for example socio-demographic data at address level or street block level, and similar for indoor networks: the number of people working along a floor could improve the analysis for evacuation plans, or the planning of floor widths etc. That means that the Network Voronoi diagram as described in this paper is the basis for many useful applications.

REFERENCES

- Aurenhammer, F., 1991: Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23 (3): 345-405.
- Aurenhammer, F.; Klein, R., 2000: Voronoi diagrams. In: Sack, J.-R.; Urrutia, J. (Eds.), *Handbook of Computational Geometry*, Elsevier Science Publishing, Amsterdam, pp. 201-290.
- de Berg, M.; van Kreveld, M.; Overmars, M.; Schwarzkopf, O., 2000: *Computational Geometry*. Springer, Berlin, 367 pp.
- Dijkstra, E.W., 1959: A note on two problems in connexion with graphs. *Numerische Mathematik*, 1: 269-271.
- Erwig, M., 2000: The Graph Voronoi Diagram with Applications. *Networks*, 36 (3): 156-163.
- Okabe, A.; Boots, B.; Sugihara, K.; Chiu, S.N., 2000: *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, United Kingdom.
- Okabe, A.; Okunuki, K.; Funamoto, S., 2002a: SANET: A Toolbox for Spatial Analysis on a Network, Center for Spatial Information Science, University of Tokyo, <http://okabe.t.u-tokyo.ac.jp/okabelab/atsu/sanet/sanet-index.html>.
- Okabe, A.; Okunuki, K.; Funamoto, S.; Ishitomi, T., 2002b: A Toolbox for Spatial Analysis on a Network and its Software. In: Zavala, G. (Ed.), *GIScience 2002*. University of California Regents, Boulder, CO, pp. 124-126.