Computers, Environment and Urban Systems 33 (2009) 90–99

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/compenvurbsys

A generic triangle-based data structure of the complete set of higher order Voronoi diagrams for emergency management

Ickjai Lee *, Kyungmi Lee

School of Business, James Cook University, Cairns Campus, McGregor Rd, Smithfield, Cairns, QLD 4870, Australia

ARTICLE INFO

Keywords:
Delaunay triangle-based data structure
Higher order Voronoi diagram
Emergency management
What-if analysis
Order-*k* Voronoi diagram
*k*th nearest Voronoi diagram
Ordered order-*k* Voronoi diagram

ABSTRACT

We introduce a generic Delaunay triangle-based data structure for geoinformation processing in disaster and emergency management. The data structure supports the complete set of higher order Voronoi diagrams (order-*k*) Voronoi diagrams, ordered order-*k* Voronoi diagrams, and *k*th nearest Voronoi diagrams for all (*k*). It provides useful and insightful information for what-if nearest queries, what-if neighboring queries, what-if zoning queries, what-if facility locating queries and what-if routing queries to handle various scenarios in the four stages of emergency management (mitigation, preparedness, response and recovery). We also demonstrate how the complete set of higher order Voronoi diagrams can be used for each phase of emergency management in diverse geoinformatics environments.

1. Introduction

Sensors monitoring regions of interest are continuously producing geospatial data for disaster analysis and emergency management. The ubiquitous data collection requires dynamic geoinformation processing for real-time emergency analysis and management. Disasters and emergencies are detrimental to people, property and environment. Real-time analysis and management of emergencies is of great importance as the lack of appropriate emergency management could lead to environmental damage, financial and structural losses or destruction of civilian infrastructure. This has become increasingly important in a time of escalating emergency situations resulting from terrorist attacks and the effects of global warming. Intelligent processing of geoinformation for emergency management is in high demand to assist with the protection of people, property and the environment from these types of emergencies.

GIS (Geographic Information Systems) providing data acquisition, interpretation and dissemination are essential in most aspects of natural disaster and emergency management (Goodchild, 2006). GIS provide a centralized mechanism to visually display emergency information (Johnson, 1994), and they have been used by many initiatives and researchers for hazard and disaster decision support (Chang, Wei, Tseng, & Kao, 1997; Dymon & Winter, 1993; Kevany, 2003; Montoya, 2003; Salt & Dunsmore, 2000). However, much of the research is limited to producing cartographic mappings and visualization rather than disaster analysis and predictive modeling (Zerger & Smith, 2003). Particularly in emergency management, (1) ordered and unordered geometric *k*-nearest queries (returning geometrically *k* nearest neighbors), (2) *k*th-order topological neighboring queries (returning *k*th-order topological neighboring regions), (3) *k*-nearest zoning (districting) queries, (4) facility locating queries, and (5) routing queries are of great importance. Currently GIS do not systemically support all of these queries simultaneously, and also lack a generic data structure supporting all these queries for "what-if" analysis in highly dynamic environments.

Geospatial tessellations attempt to answer some of these queries by segmenting the space into meaningful sub-regions (districts) (Okabe, Boots, Sugihara, & Chiu, 2000). For instance, they have been used in market area analysis and spatial competition (Hanjoul, Beguin, & Thill, 1989; Okabe et al., 2000; Rushton & Thill, 1989; Thill & Rushton, 1992). Hanjoul et al. (1989) investigated the boundaries of market areas by tessellating the space into regions of dominance in the Euclidean space. Rushton and Thill (1989) and Thill and Rushton (1992) further compared spatial competition and the boundaries of market areas in the Euclidean space against the Manhattan space. However, these studies are limited to order-1 (dominant region of one city) and are not able to model higher order dominant regions such as a dominant region of four cities (order-4). In addition, these studies theoretically model the boundaries of market areas, but do not provide a data structure for practical implementation. The Voronoi diagram and its dual Delaunay triangulation offer a robust framework for modeling and structuring geospatial tessellations (Okabe et al., 2000). They are used for what-if analysis in a wide spectrum of geosciences and environmental sciences (Okabe et al., 2000). Higher order Voronoi diagrams, popular generalizations of the Voronoi diagram, provide

* Corresponding author.
E-mail addresses: Ickjai.Lee@jcu.edu.au (I. Lee), Joanne.Lee@jcu.edu.au (K. Lee).

informative insights into the generalized what-if queries suggested previously. Despite the popularity of higher order Voronoi diagrams, we still lack a generic unified data structure to support them.

This paper introduces a generic Delaunay triangle-based data structure for geoinformation processing in disaster and emergency management. For a given set of n generators, this data structure encompasses the complete set of higher order Voronoi diagrams (referred to as the complete higher order Voronoi diagrams) providing useful information for nearest, neighboring, zoning, locating and routing queries to handle various scenarios in the four stages of emergency management (mitigation, preparedness, response, and recovery).

The complete higher order Voronoi diagrams are all order-k, ordered order-k Voronoi diagrams, and kth nearest Voronoi diagrams for $k = 1, \dots, n-1$. The framework builds a unified order-k Delaunay triangle data structure from which users can derive the complete higher order Voronoi diagrams for various emergency scenarios to support various what-if scenarios. Building the generic data structure requires $O(n^4)$ time, and once it is determined computing the complete higher order Voronoi diagrams requires $O(n^3 \log n)$ time. In this article, the unified data structure is limited to the Euclidean space where "crow flies distance" applies. This is of particular use in large scale analysis, and marine or aerial environments where a straight line between two targets makes sense.

2. A generic Delaunay triangle-based data structure for emergency management

2.1. Background

Given a set $P = \{p_1, p_2, \dots, p_n\}$ of distinct generators, the ordinary Voronoi diagram is obtained by assigning all locations in the space to the closest generator. Locations equidistant from two generators form Voronoi edges while locations equidistant from more than two generators constitute Voronoi vertices. This assignment tessellates the space into mutually exclusive and collectively exhaustive Voronoi regions. Higher order Voronoi diagrams (order-k, ordered order-k, and kth nearest Voronoi diagrams) are popular generalizations of the ordinary Voronoi diagram to more than one generator. They provide space tessellations where each point in a region has the same k (ordered or unordered) closest generators for a given k. These tessellations are useful for situations where there may be more than one location of interest, some are mal-functioning (busy, closed, inaccessible, unavailable or fully scheduled) or some close neighbors are required to collaborate to quickly complete a given task. These situations occur regularly in emergency management and so the application of the complete higher order Voronoi diagrams offers a useful tool for emergency management.

Suppose we are given P in $\mathbb{R}^2$ where $2 \leqslant n < \infty$. The order-k Voronoi diagram $\mathcal{V}^{(k)}$ is a set of all non-empty order-k Voronoi regions $\{V(P_1^{(k)}), \dots, V(P_l^{(k)})\}$, where the non-empty order-k Voronoi region $V(P_i^{(k)})(V(P_i^{(k)}) \neq \emptyset)$ for a certain subset $P_i^{(k)}$ consisting of k points out of P defined as:

$$V(P_i^{(k)}) = \{p | \max_{p_r \in P_i^{(k)}} d(p, p_r) \leqslant \min_{p_s \in P \setminus P_i^{(k)}} d(p, p_s)\}.$$

Note that the number of non-empty order-k Voronoi regions is $O(k(n-k))$ and the actual size of l is $O(n^2)$ in $\mathbb{R}^2$ (Lee, 1982).

In the order-k Voronoi diagram, k generators are not ordered, however in some situations an ordered set (sequence) would be of interest. This can be modeled by the ordered order-k Voronoi diagram $\mathcal{V}^{<k>}$. It is a set of non-empty ordered order-k Voronoi regions $\{V(P_1^{<k>}), \dots, V(P_m^{<k>})\}(V(P_i^{<k>}) \neq \emptyset)$, where $m = n(n-1)\dots(n-k+1)$ and the non-empty ordered order-k Voronoi region of $P_i^{<k>} = \{p_{i1}, \dots, p_{ik}\}$ defined as:

$$V(P_i^{<k>}) = \{p | d(p, p_{i1}) \leqslant \dots \leqslant d(p, p_{ik}) \leqslant d(p, p_j),$$
$$p_j \in P \setminus \{p_{i1}, \dots, p_{ik}\}\}.$$

$\mathcal{V}(P_i^{<k>})$ is a refinement of $\mathcal{V}(P_i^{(k)})$, namely $V(P_i^{(k)}) = \bigcup_{P_j^{<k>} \in A^{<k>}(P_i^{(k)})} V(P_j^{<k>})$, where $A^{<k>}(P_i^{(k)})$ is the sequence of all possible k-tuples made of $p_{i1}, \dots, p_{ik}$ (Okabe et al., 2000).

One generalized variant of the ordinary Voronoi diagram similar to $\mathcal{V}^{(k)}$ is the kth nearest Voronoi diagram. This is particularly useful when users are interested in only kth nearest region. The kth nearest Voronoi diagram $\mathcal{V}^{[k]}$ is a set of all kth nearest Voronoi regions $\mathcal{V}^{[k]} = \{V^{[k]}(p_1), \dots, V^{[k]}(p_n)\}$, where the kth nearest Voronoi region $V^{[k]}(p_i)$ is defined as:

$$V^{[k]}(p_i) = \{p | d(p, p_i) \leqslant d(p, p_j),$$
$$p_j \in P \setminus \{k \text{nearest generators of } p_i\}\}.$$

$\mathcal{V}(P_i^{<k>})$ is a refinement of $\mathcal{V}(P_i^{[k]})$. That is, $V(P_i^{[k]}) = \bigcup_{(p_{j1}, \dots, p_{jk-1}) \in A^{<k-1>}(P \setminus \{p_i\})} V((p_{j1}, \dots, p_{jk-1}, p_i))$, where $A^{<k-1>}(P \setminus \{p_i\})$ is the set of all possible $(k-1)$-tuples consisting of $k-1$ elements out of $P \setminus \{p_i\}$ (Gahegan & Lee, 2000; Lee & Gahegan, 2002; Okabe et al., 2000).

Fig. 1 illustrates higher order Voronoi diagrams with a set P of five generators, $\{p_3, p_4, p_5, p_6, p_7\}$. Fig. 1a depicts the ordinary Voronoi diagram of P. The shaded Voronoi region in Fig. 1a is $V(p_7)$ having p_7 as the closest generator. Fig. 1b displays the order-3 Voronoi diagram of $P, \mathcal{V}^{(3)}(P)$. A shaded region is $V(P^{(3)}_{\{p_3,p_4,p_7\}})$ having three generators $\{p_3, p_4, p_7\}$ as the first three closest. Fig. 1c shows the ordered order-3 Voronoi diagram of $P, \mathcal{V}^{<3>}(P)$. The shaded region in Fig. 1b is further decomposed into six $(3 * 2 * 1)$ exclusive ordered order-3 Voronoi regions as shown in Fig. 1c. Fig. 1d depicts the 3rd-nearest Voronoi diagram of $P, \mathcal{V}^{[3]}(P)$. The shaded region in Fig. 1b is further decomposed into three 3rd-nearest Voronoi regions. More details can be found in (Aurenhammer & Schwarzkopf, 1991; Okabe et al., 2000).

2.2. Problem statement and motivation

Even though higher order Voronoi diagram families $\mathcal{V}^{(k)}, \mathcal{V}^{<k>}$, and $\mathcal{V}^{[k]}$ are useful and highly correlated, little research has been conducted on structural arrangements of these families (Gahegan & Lee, 2000). Several algorithmic approaches have been proposed to efficiently compute $\mathcal{V}^{(k)}$ (Aurenhammer & Schwarzkopf, 1991; Dehne, 1983; Lee, 1982). These are categorized into two groups: with data structure and without data structure. As an example of the former (with data structure), Dehne (1983) proposed an $O(n^4)$ time algorithm based on a Delaunay triangle-based data structure that constructs the complete order-k Voronoi diagrams. Several other attempts (Aurenhammer & Schwarzkopf, 1991; Chazelle & Edelsbrunner, 1987; Lee, 1982) have been made without the Delaunay triangle-based data structure in order to improve the computational time requirement of Dehne's algorithm. These (without data structure) directly compute the order-k Voronoi diagrams that have difficulty supporting topological neighboring queries. Dehne's Delaunay triangle-based data structure is a robust candidate for geospatial applications since it supports both geometrical and topological neighborhoods. However, its time efficiency is comparatively expensive and it only supports the complete order-k Voronoi diagrams. In the literature, interrelationships of higher order Voronoi diagram families ($\mathcal{V}^{(k)}, \mathcal{V}^{<k>}$, and $\mathcal{V}^{[k]}$) and applications to the real-world problem have attracted little attention (Okabe, Boots, & Sugihara, 1994). Currently, the best known algorithm for the order-k Voronoi diagram is $O(k(n-k)\log n + n\log^3 n)$

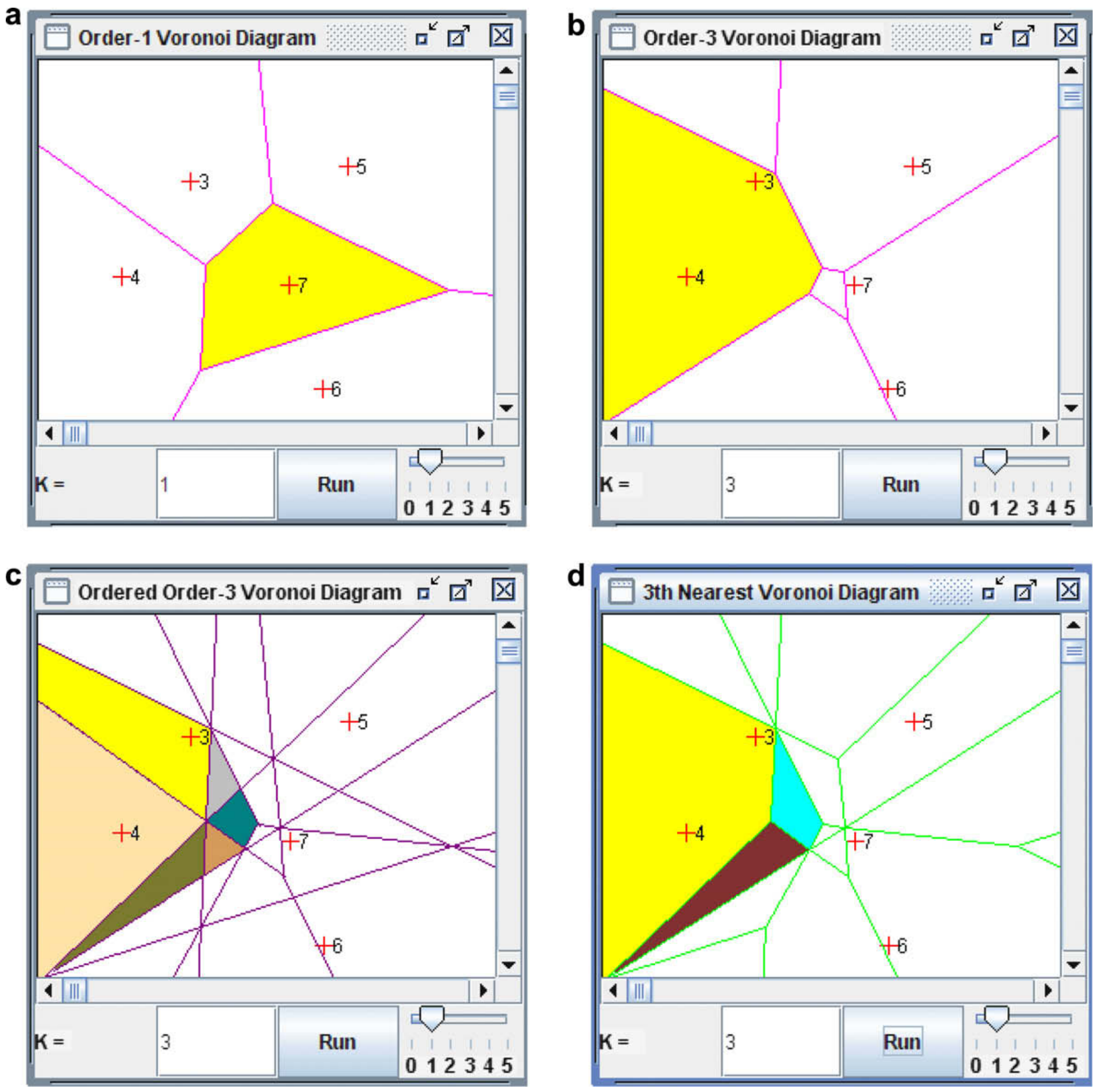

Fig. 1. Higher order Voronoi diagrams ($|P| = 5$) where $p_i \in P$ is depicted as a cross (+) and an associated identification number i : (a)$\mathcal{V}^{(1)}(P)$; (b) $\mathcal{V}^{(3)}(P)$; (c) $\mathcal{V}^{<3>}(P)$; (d) $\mathcal{V}^{[3]}(P)$.

(de Berg, van Kreveld, Overmars, & Schwarzkopf, 2000), and thus the complete order-k Voronoi diagrams requires $O(n^3 \log n)$ time.

In this article, we improve this Delaunay triangle-based data structure and provide algorithms computing the complete order-k, ordered order-k and k-nearest neighbor Voronoi diagrams in $O(n^3 \log n)$ time when the generic data structure is determined (details will be discussed in Section 2.5.3). We explain how these complete higher order Voronoi diagrams can be derived from a unified Delaunay triangle-based data structure, how they can be used to support the nearest, neighboring, zoning, locating and routing queries, and provide examples how they can be used for emergency management. We analyze time complexity and space requirements of the proposed method.

2.3. Framework and functionality

Fig. 2 depicts the framework of our approach. The unified data structure may bulk-load input data and can allow for dynamic updates (additions and deletions). What-if queries support for instant exploration of scenarios when changes are made. Once our proposed data structure is computed, it enables users to retrieve the complete higher order Voronoi diagrams and various topological information without reconstruction. For instance, given P (a set of objects of interest such as a set of distributed sensors, emergency units, or disasters) within a study region S, the data structure will provide useful geometrical and topological information for a target location $p \in S$ for the following what-if queries:

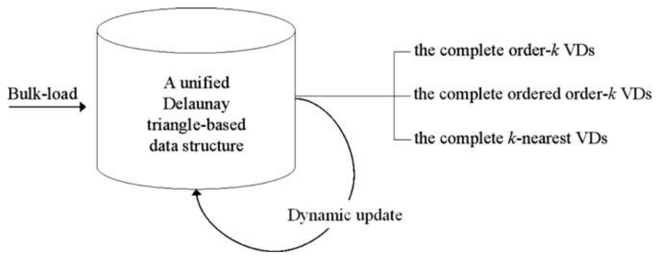
Bulk-load
A unified Delaunay triangle-based data structure
the complete order-k VDs
the complete ordered order-k VDs
the complete k-nearest VDs
Dynamic update

Fig. 2. The framework of our approach.

- geometric nearest neighbor queries:
  - the kth-nearest object for p;
  - a set of ordered k-nearest objects for p;
  - a set of unordered k-nearest objects for p;

- topological neighboring queries:
  - the kth-order neighboring regions for p (A region is a kth-order neighbor of a given region if the region is an immediate neighbor of one $(k-1)$-order neighbor of the given region. Two regions are 1st-order neighbors if they share a common boundary of Voronoi region.);

- zoning (segmenting space into mutually exclusive and collectively exhaustive regions) queries:
  - the kth-nearest zone (segmented region) of p to which p is the kth-nearest;
  - the order-k zone of p in which p is one of unordered k-nearest neighbors;
  - the ordered order-k zone of p in which p is the k-nearest in the ordered list;

- facility locating queries:
  - locations optimizing a given need;

- routing queries:
  - routes crossing Voronoi edges that maximize distances from objects (particularly disasters).

What-if nearest queries are of great importance in geospatial analysis and in emergency management (Okabe et al., 1994). For instance, residents might be interested in the nearest order-k emergency units or in the ordered order-k disasters. These types of queries reveal geometric neighborhood relations. Alternatively, what-if neighboring queries can be used to uncover topological neighborhood relations returning kth-order adjacent Voronoi regions. When k is one, it returns adjacent Voronoi regions to a given target. When k becomes two, it finds adjacent Voronoi regions adjacent to the 1st-order Voronoi regions. What-if zoning queries district S into meaningful zones (areas of influence) while what-if facility locating queries optimize facility locations for a given need. What-if routing queries are of particular use in evacuation plan finding a path to a target evacuation shelter while minimizing danger factors (typically distances to neighboring disasters). The complete higher order Voronoi diagrams provide a solid framework for these queries. Higher order Voronoi regions are informative of what-if nearest queries, neighboring queries, zoning queries, facility locating queries, and routing queries. Voronoi vertices are suggestive of what-if facility locating queries since they are the gravity centers from neighboring objects. Higher order Voronoi edges are informative of what-if routing queries since they are the farthest paths from the nearest neighboring disasters.

2.4. A unified Delaunay triangle-based data structure

Our unified Delaunay triangle-based data structure is the combination of two lists: generator list and triangle lists, which is similar to the triangle-based data structure (Dehne, 1983; Lee & Gahegan, 2002). The generator list stores points containing the x, y coordinate with a unique identification number. The triangle lists store triangles, each with three vertices, a circumcircle center point and a list of generator identifications lying within the circumcircle.

In the unified data structure, the triangles are assigned to the triangle list based on the number of generators that are lying within the circumcircle for this triangle. These triangles are referred to as order-k triangles, where k is the number of generators within the circumcircle of this triangle. Table 1 depicts the assignment of triangles shown in Fig. 3. The triangle $\Delta p_1p_2p_5$ does not contain any other points in its circumcircle, thus it is an order-0 triangle. In contrast, $\Delta p_1p_2p_4$ contains one point (p_5) within its circumcircle, thus it is an order-1 triangle. In a similar way, $\Delta p_1p_2p_3$ contains two points (p_4, p_5) within its circumcircle, thus it is an order-2 triangle.

2.5. Dynamic updates in the unified data structure

When dynamic updates (insertions or deletions) occur, the unified data structure has to handle two types of triangles. For a chosen p for insertion or deletion, triangles with p inside their circumcircles and triangles with p as one of the vertices must be updated. Insertion firstly updates triangles that contain p (*Algorithm* **Triangle Update**) and generates new triangles having p as one of the vertices (*Algorithm* **Generate New Triangles**). Deletion is the reverse of insertion, iterating through all triangles and removing any triangles having p as a vertex and any triangles containing p. The algorithms used are discussed in more detail below.

2.5.1. Insertion

All the existing triangles in the data structure are checked to determine whether their circumcircles contain the new generator p. If they do then order-k triangles become order-$(k+1)$ triangles, and the corresponding inpoints are updated.

Algorithm. Triangle Update

```
Input: A newly added point $p$ and the unified data structure ($DS$);
Output: Updated unified data structure;
(1) begin
(2) $i := 0$
(3) do
(4)     $\Delta abc := DS[i]$
(5)     if (isInCircle($\Delta abc, p$))
(6)       $\Delta abc$.addInPoint($p$)
(7)     end if
(8)     $i := i + 1$
(9) while ($i <| DS |$)
(10) end
```

Here $DS[i]$ denotes ith data object in the triangle list DS, `isInCircle` $(\Delta abc, p)$ checks if the circumcircle of Δabc contains p within it and returns a boolean value, and Δabc.`addInPoint` (p) adds p to the `Inpoint`list of Δabc.

Once the existing triangles are updated, new triangles for p can be generated. The algorithm shown in the pseudocode below iterates through all the possible unordered pairs of $\{p_1, p_2, \ldots, p_n\}$ and creates a triangle with p as the third vertex.

Table 1
Complete order-k Delaunay triangle data structure (onpoint: three points of Δ; inpoint: points contained within the circumcircle of Δ).

Order-k Delaunay triangle	Triangle (Δ) ID	Onpoint Δabc	Inpoint
Order-0 Delaunay triangle	1	$\{p_1, p_2, p_5\}$	ϕ
Order-0 Delaunay triangle	2	$\{p_2, p_3, p_5\}$	ϕ
Order-0 Delaunay triangle	3	$\{p_3, p_4, p_5\}$	ϕ
Order-0 Delaunay triangle	4	$\{p_1, p_4, p_5\}$	ϕ
Order-1 Delaunay triangle	5	$\{p_1, p_2, p_4\}$	$\{p_5\}$
Order-1 Delaunay triangle	6	$\{p_2, p_3, p_4\}$	$\{p_5\}$
Order-2 Delaunay triangle	7	$\{p_1, p_2, p_3\}$	$\{p_4, p_5\}$
Order-2 Delaunay triangle	8	$\{p_1, p_3, p_4\}$	$\{p_2, p_5\}$

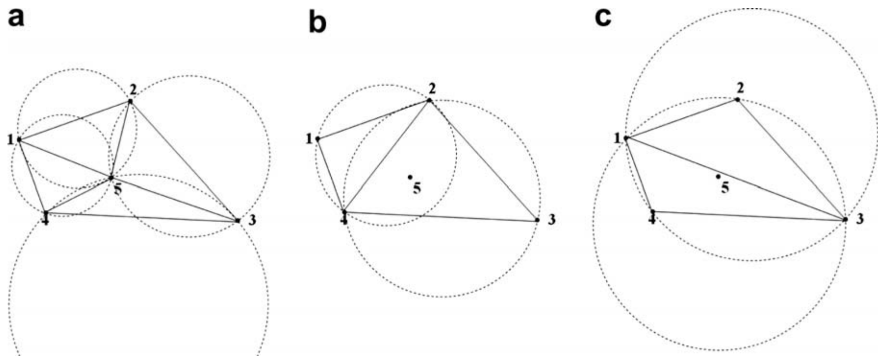

Fig. 3. Circumcircles and data structure for the complete higher order Voronoi diagrams: (a) order-0 triangles; (b) order-1 triangles; (c) order-2 triangles.

Algorithm. Generate New Triangles

```
Input: A newly added generator $p$, a set of generators
    $\{p_1, p_2, \dots, p_n\}$ and the unified data structure ($DS$);
Output: Updated unified data structure;
(1) begin
(2) $i := 0$
(3) while ($i < n$)
(4)     $j := i + 1$
(5)     while($j < n$)
(6)       $\Delta pp_ip_j$.addInPoint(FindInPoints ($\Delta pp_ip_j$))
(7)       $k :=$ getNumberOfInPoints ($\Delta pp_ip_j$)
(8)       SaveToDS ($\Delta pp_ip_j, k$)
(9)       $j := j + 1$
(10)    $i := i + 1$
(11) end
```

Each triangle is first evaluated with the FindInPoints() which linearly iterates through generators and looks for those that are within the triangle's circumcircle. These are added to the InPoints in Step 6. Once all the InPoints are found, the triangle is then saved to the corresponding order-k triangle table based on the number of generators in InPoints list (SaveToDS).

2.5.2. Deletion

The deletion approach finds both triangles that have p as one of vertices and whose circumcircles contain p. For the first case, the triangle is removed from DS. For the second case, the triangles are updated from order-k triangles to order-$(k-1)$ triangles.

Algorithm. Deletion

```
Input: A deleted point $p$ and the unified data structure ($DS$);
Output: Updated unified data structure;
(1) begin
(2) $i := 0$
(3) do
(4)     $\Delta abc := DS[i]$
(5)     if ($p == a$ or $p == b$ or $p == c$)
(6)       $k :=$ getNumberOfInPoints ($\Delta abc$)
(7)       RemoveDS($\Delta abc, k$)
(8)     end if
(9)     if (isInCircle($\Delta abc, p$))
(10)        $k :=$ getNumberOfInPoints ($\Delta abc$)
(11)        $\Delta abc$.removeInPoint($p$)
(12)        RemoveDS($\Delta abc, k$)
(13)        SaveDS($\Delta abc, k - 1$)
(14)    end if
(15)    $i := i + 1$
(16) while ($i <| DS |$)
(17) end
```

In dynamic updates, $O(n^2)$ order-k triangles need $O(n)$ time to update. Thus, dynamic updates requires $O(n^3)$ time while computing the complete order-k Delaunay triangle data structure itself requires $O(n^4)$ time.

2.5.3. Derivation of the complete higher order Voronoi diagrams from the unified data structure

Derivation of the complete order-k Voronoi diagrams from the unified data structure is based on the fact that the center of three onpoints of an order-k triangle is a Voronoi vertex of $\mathcal{V}^{(k+1)}$ and $\mathcal{V}^{(k+2)}$ (Dehne, 1983). That is, $\mathcal{V}^{(k)}$ is derived from the combination of both order-$(k-1)$ and order-$(k-2)$ triangles. $\mathcal{V}^{(1)}$ is derived from the order-0 triangle list while $\mathcal{V}^{(2)}$ is derived from the order-0 and order-1 triangle lists. For each order-k triangle ($\triangle ABC$), we can draw three lines from the center (M) of ABC perpendicular to each line segment of any two onpoints ($\overline{AB}, \overline{BC}, \overline{CA}$) (Compute step). These lines are half portions of bisectors. They form $\mathcal{V}^{(k+1)}$ Voronoi edges while the other half portions of bisectors form $\mathcal{V}^{(k+2)}$ Voronoi edges (Dehne, 1983). Once we have all these bisectors computed, then we can combine neighboring portions of bisectors together to derive the complete order-k Voronoi diagrams (Merge step).

Note that, the number of triangles in the data structure is $O(n^3)$ and the number of order-k triangles is $O(n^2)$ (Shamos & Hoey, 1975). Each order-k triangle is associated with k inpoints. Differently from Dehne (1983)'s data structure, our data structure uses a deterministic hash function to store associated inpoints. The hash function uses $O(n)$ space and guarantees $O(\log n)$ insertion, lookup and deletion time (Mehlhorn & Näher, 1999). The use of hash function decreases memory requirement, but increases computational time. Our data structure requires $O(n^3)$ memory, but each order-k triangle requires $O(\log n)$ time to process. That is, computing portions of bisectors for all order-k triangles requires $O(n^3)$ time to iterate order-k triangles and $O(\log n)$ time to process each triangle. Thus, it requires $O(n^3 \log n)$ time to compute portions of bisectors. Also, for each order-k triangle, it needs to find neighboring order-k triangles that can be achieved within $O(\log n)$ time with geospatial indexing data structures (Samet, 2006; Shamos & Hoey, 1975). That is, the nearest neighboring problem can be solved in $O(\log n)$ time using geospatial indexing data structures, thus finding neighboring order-k triangles for each order-k triangles requires logarithmic time. Therefore, Merge step requires $O(n^2 \log n)$ for each k and it requires $O(n^3 \log n)$ time for all k.

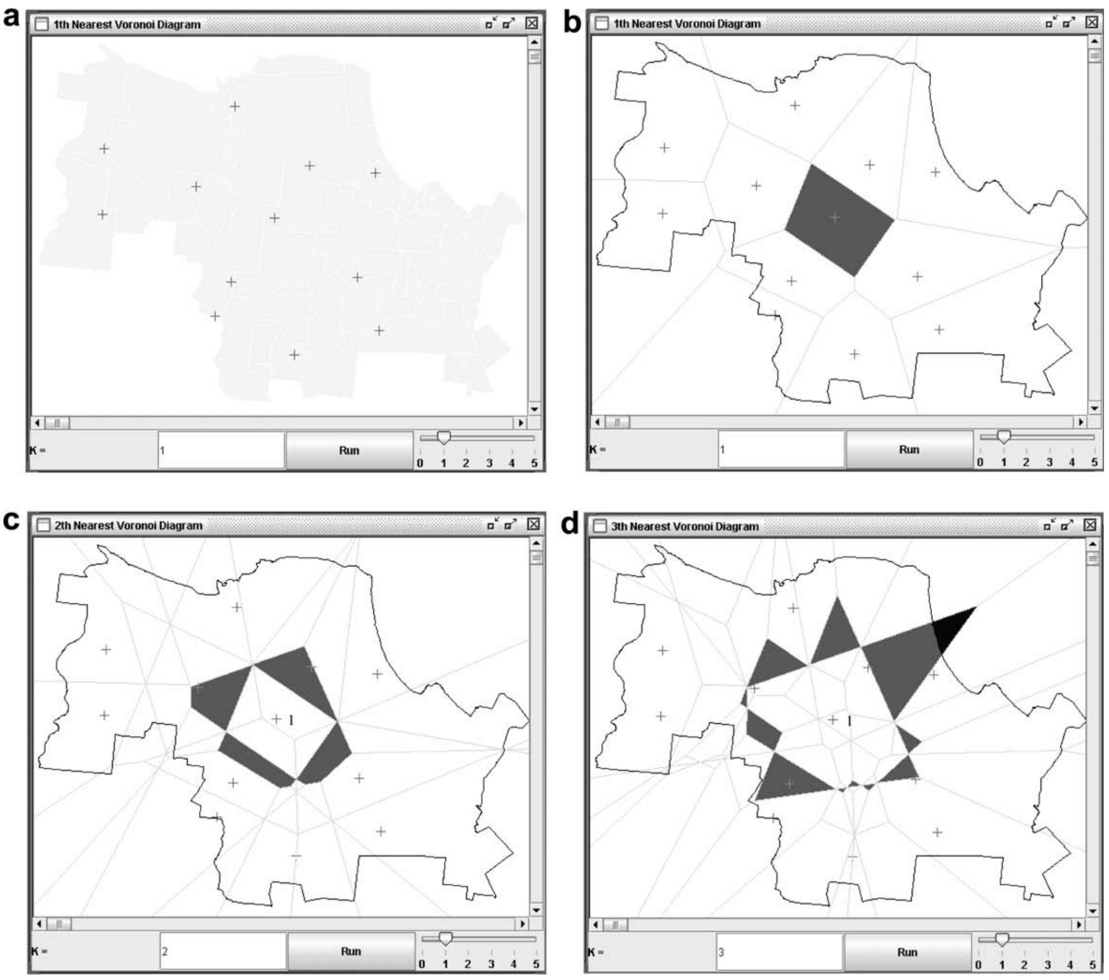
a
1th Nearest Voronoi Diagram
K =
1
Run
0 1 2 3 4 5
b
1th Nearest Voronoi Diagram
K =
1
Run
0 1 2 3 4 5
c
2th Nearest Voronoi Diagram
1
K =
2
Run
0 1 2 3 4 5
d
3th Nearest Voronoi Diagram
1
K =
3
Run
0 1 2 3 4 5

Fig. 4. $\mathscr{V}^{[k]}(P)$ ($|P| = 12$) where $p_i \in P$ is depicted as a cross (+): (a) S with P; (b) $\mathscr{V}^{[1]}(P)$; (c) $\mathscr{V}^{[2]}(P)$; (d) $\mathscr{V}^{[3]}(P)$.

Once the complete $\mathscr{V}^{(k)}$ is computed, $\mathscr{V}^{<k>}$ and $\mathscr{V}^{[k]}$ can be computed from the following lemmas. Therefore, the complete higher order Voronoi diagrams can be computed with $O(n^3 \log n)$ time from the complete order-k Delaunay triangle data structure. Note that when $k = 1, \mathscr{V}^{(1)} = \mathscr{V}^{<1>} = \mathscr{V}^{[1]}$.

Lemma 1. $\mathscr{V}^{<k>} = \mathscr{V}^{<k-1>} \cup \mathscr{V}^{(k)}$, for $k = 2, \ldots, n-1$.

Proof. Note that, $\mathscr{V}(P_i^{<k>})$ is a refinement of $\mathscr{V}(P_i^{(k)})$, namely $V(P_i^{(k)}) = \bigcup_{P_j^{<k>} \in A^{<k>}(P_i^{(k)})} V(P_j^{<k>})$, where $A^{<k>}(P_i^{(k)})$ is the set of all possible k-tuples made of $p_{i1}, \ldots, p_{ik}$ (Okabe et al., 2000). That is, the order-k Voronoi region $V(P_i^{(k)})$ needs to be further decomposed into sub-regions based on the order of k. Since $\mathscr{V}^{<k-1>}$ tessellates the space into ordered order-$(k-1)$ Voronoi regions and defines the order of k, the union of $\mathscr{V}^{<k-1>}$ and $\mathscr{V}^{(k)}$ formulates $\mathscr{V}^{<k>}$. □

Lemma 2. $\mathscr{V}^{[k]} = \mathscr{V}^{(k-1)} \cup \mathscr{V}^{(k)}$, for $k = 2, \ldots, n-1$.

Proof. Note that, $\mathscr{V}(P_i^{<k>})$ is a refinement of $\mathscr{V}(P_i^{[k]})$. That is, $V(P_i^{[k]}) = \bigcup_{(p_{j1},\ldots,p_{jk-1}) \in A^{<k-1>}(P \setminus \{p_i\})} V((p_{j1}, \ldots, p_{jk-1}, p_i))$, where $A^{<k-1>}(P \setminus \{p_i\})$ is the set of all possible $(k-1)$-tuples consisting of $k-1$ elements out of $P \setminus \{p_i\}$ (Okabe et al., 2000). Thus, $\mathscr{V}^{[k]}$ can be computed once $\mathscr{V}(P_i^{<k>})$ is constructed. □

3. Emergency management with higher order Voronoi diagrams

Emergency management is composed of four basic phases: mitigation, preparedness, response, and recovery (Haddow & Bullock, 2003). This section demonstrates how the complete higher order Voronoi diagrams can be used for each stage of emergency management. We explain this with a study region S of Townsville, a tropical city of northern Queensland in Australia. The study region consists of 42 urban suburbs of Townsville and suffers from periodic tropical cyclones accompanying strong winds and heavy rains.

3.1. Mitigation

Mitigation is a realization of the causes of risk/danger, and is a factor that applies to preventing (removing, eliminating, reducing) disasters (emergencies) (Haddow & Bullock, 2003; Johnson, 1994). Tropical cyclones are extremely destructive as a result of strong winds, flooding and storm surges. Mitigation plays a crucial role in this area and identifying the vulnerability of people, property and environment is of upmost importance. Note that disasters occur when hazards meet vulnerability (Wisner, Blaikie, Cannon, & Davis, 2003). Information could be collected using sensors to identify vulnerable targets and allow emergency personnel to take appropriate actions. In this mitigation process, identifying (ordered and unordered) k-nearest points and zones is important in planning and decision-making. In the following example, we take the what-if nearest, neighboring and zoning queries.

Fig. 4 demonstrates $\mathscr{V}^{[k]}$ of locations of 12 sensors within S. Fig. 4b shows $\mathscr{V}^{[1]}$, Fig. 4c shows $\mathscr{V}^{[2]}$ while Fig. 4d depicts $\mathscr{V}^{[3]}$. k-nearest zones of a particular sensor indicate its regions of interest or its territories to check for vulnerability. The shaded region in Fig. 4b contains p_1. Within this region, p_1 is the first nearest sensor

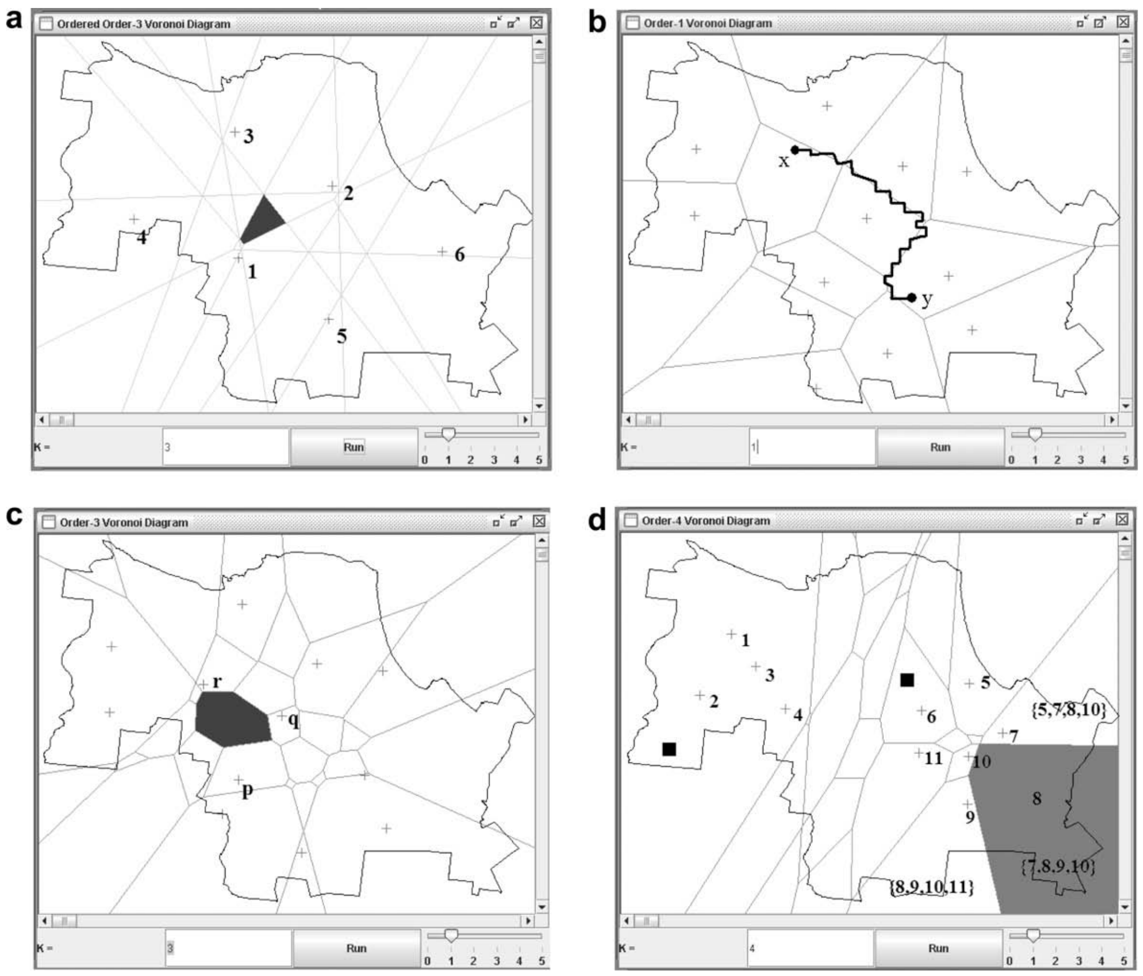

Fig. 5. Various higher order Voronoi diagrams: (a) $\mathscr{V}^{<3>}(P)$ ($|P| = 6$) where $p_i \in P$ is depicted as a cross (+) and an associated identification number i; (b) $\mathscr{V}^{(1)}(P)$ (P same as in Fig. 4); (c) $\mathscr{V}^{(3)}(P)$ (P same as in Fig. 4); (d) $\mathscr{V}^{(4)}(P)$($|P| = 11$) where $p_i \in P$ is shown as a cross (+) and an associated number i.

and so p_1 is primarily in charge of this region. Shaded regions in Fig. 4c are regions having p_1 as the second nearest sensor indicating sensor p_1 should be supporting or on standby just in case those areas not covered by their first nearest sensors. Shared regions in Fig. 4d encompass the first and second nearest Voronoi regions and indicate the next level involvement. The complete $\mathscr{V}^{[k]}$ is useful for a comprehensive coverage of mitigation activities.

3.2. Preparedness

Preparedness involves making plans to minimize damage such as having emergency personnel on standby or determining the nearest evacuation locations. This requires information on the ordered k nearest emergency personnel or the ordered k nearest evacuation locations. This can be modeled by the ordered order-k Voronoi diagram and Fig. 5a depicts $\mathscr{V}^{<3>}(P)$ where $P = \{p_1, p_2, \cdots, p_6\}$. The shaded region has p_1, p_2, and p_3 as the first, second and third nearest, respectively.

If we assume that P represents evacuation shelters then people living in the shaded region will have p_1 as the nearest evacuation shelter, p_2 as the second and p_3 as the third. Thus, people living in this shaded region will have a clear understanding of where they should go when the first shelter is unavailable, when the first is fully booked or roads to the first shelter are damaged. Higher order $\mathscr{V}^{<k>}$ can be in place when higher k is required. The shaded region is also useful for what-if facility location queries. When P represents a set of disasters, the shaded region is suggestive of a candidate area to which a new emergency unit can be introduced to handle the first three nearest disasters p_1, p_2, and p_3.

Fig. 5b illustrates the usefulness of higher order Voronoi edges for routing in disaster management. Voronoi edges are the farthest locations from nearby neighboring objects and they are indicative of a safer pathway from a source to a target. When we travel from a source x to a target y when P occurs as in Fig. 5b, we need to find roads or streets that straddle Voronoi edges in order to minimize danger factors. That is, road networks crossing Voronoi edges can be extracted. The bold polyline is one such solution.

3.3. Response

Response consists of activities that provide emergency assistance for victims. Typically, the k-closest emergency unit is required to quickly respond to emergencies. It is also important to identify the k-nearest resources such as hydrants in urban fires, electronic pumps in floods, and ships in oil-spills. Here we use order-k Voronoi diagrams as an example.

When emergencies occur, k nearest emergency units are required to cooperate and collaborate in order to reduce damage. Depending on the type and severity of the emergency, k can be determined. The complete order-k Voronoi diagram can model this and can be used for dynamic emergency response. Fig. 5c demonstrates an example with $k = 3$. Each $\mathscr{V}^{(3)}$ Voronoi region in Fig. 5c shows an area where three emergency units need to work together,

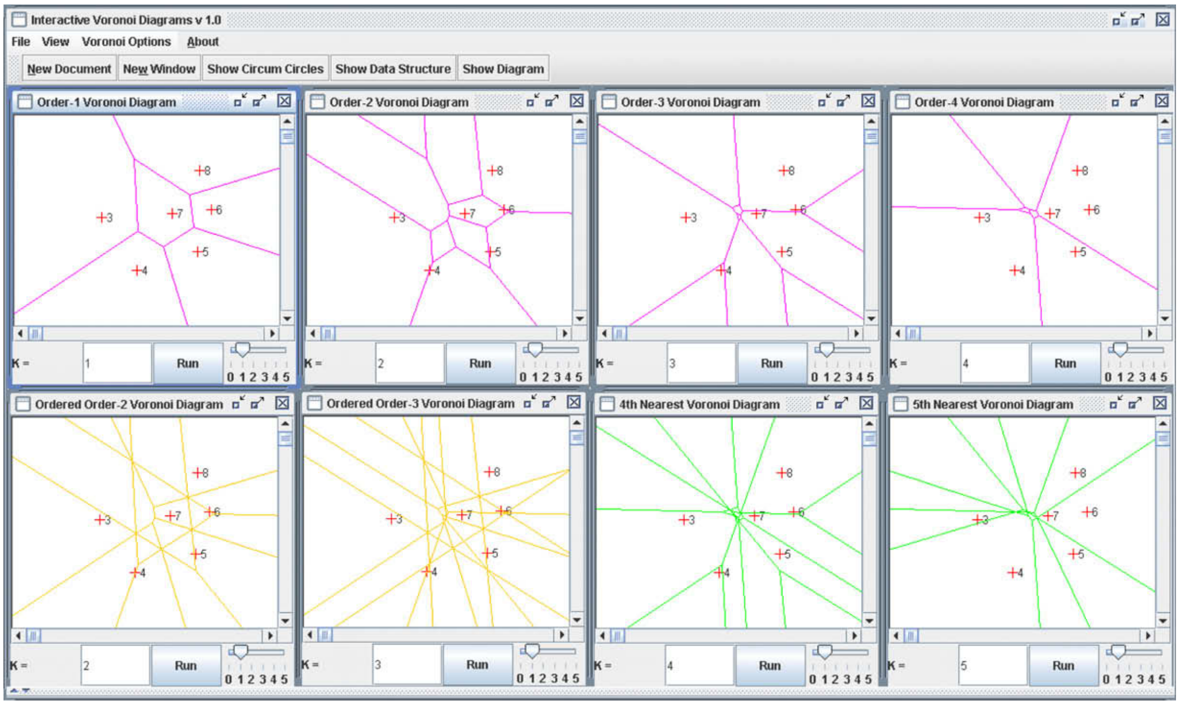

Fig. 6. $\mathscr{V}^{(1)}(P), \mathscr{V}^{(2)}(P), \mathscr{V}^{(3)}(P), \mathscr{V}^{(4)}(P), \mathscr{V}^{<2>}(P), \mathscr{V}^{<3>}(P), \mathscr{V}^{[4]}(P)$ and $\mathscr{V}^{[5]}(P)(P = \{p_3, p_4, p_5, p_6, p_7, p_8\})$ where $p_i \in P$ is shown as a cross (+) and an associated identification number i.

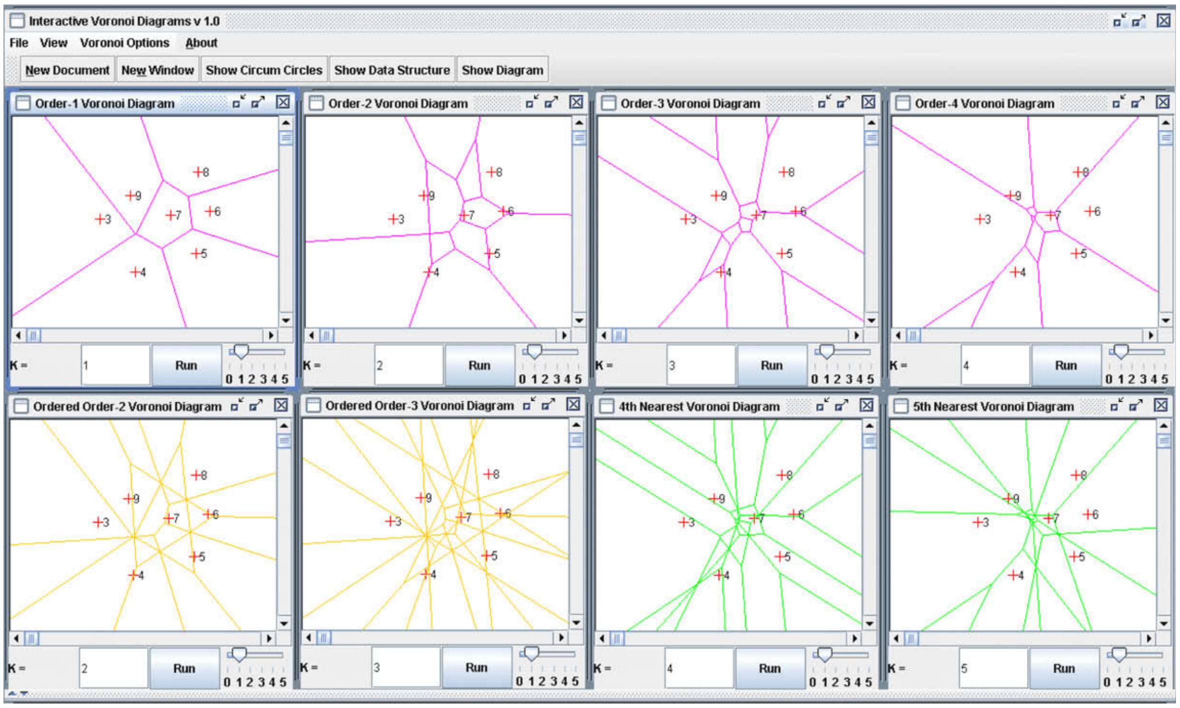

Fig. 7. $\mathscr{V}^{(1)}(P'), \mathscr{V}^{(2)}(P'), \mathscr{V}^{(3)}(P'), \mathscr{V}^{(4)}(P'), \mathscr{V}^{<2>}(P'), \mathscr{V}^{<3>}(P'), \mathscr{V}^{[4]}(P')$ and $\mathscr{V}^{[5]}(P')$ $(P' = P + \{p_9\}, P$ as in Fig. 6) where $p_i \in P$ is shown as a cross (+) and an associated identification number i.

such that p, q, and r emergency units are required to collaborate. These $\mathscr{V}^{(k)}$ diagrams help emergency management officials make prompt decisions and predictive modeling. For instance, emergency management officials can determine a candidate number suitable for k by varying it from 1 to $n-1$.

3.4. Recovery

Recovery is a series of activities that reduce the secondary damage and return damaged areas back to normal. Due to the strong winds and floods that accompany a cyclone, there is typically extensive road-damage. This road-damage is a serious cause of road accidents, and must be identified and repaired in a timely manner.

In many emergency recovery situations, one of the obstacles that hinders the recovery process is limited resources (personnel or equipments). This is particularly true when multiple emergencies are taking place simultaneously. Hence, an emergency management official needs to optimize locations of emergency personnel and has to determine the minimum number of personnel required. Fig. 5d depicts one such example. It shows 11 road-damage locations and two emergency units (denoted by rectangles) already in place.

Let us assume that each unit is able to handle only a certain number of road-damage tasks. In this case, each unit can deal with the first four closest road-damage tasks. One unit is in charge of $\{p_1, p_2, p_3, p_4\}$ and the other is in charge of $\{p_5, p_6, p_{10}, p_{11}\}$. Consequently, three road-damage locations $\{p_7, p_8, p_9\}$ are left unfixed. An emergency management official intends to introduce a new emergency unit (capacity of four) somewhere in S so that the unit can attend to the road-damage as a subset of the first four closest (assuming the Euclidean distance). In this case, the order-4 Voronoi diagram provides useful information and suggests an informative recommendation. An order-4 Voronoi region having these three road-damage locations as a subset of its generators is a region of interest. In Fig. 5d, a new emergency unit can be introduced in the shaded Voronoi region in the bottom right corner covering the set of uncovered road-damage.

3.5. Emergency management with complete higher order Voronoi diagrams

With the aid of the data structure and algorithms, we are able to derive the complete $\mathcal{V}^{(k)}$, $\mathcal{V}^{<k>}$, and $\mathcal{V}^{[k]}$ from the same dataset P. This provides a rich environment to emergency management officials that enables them to explore what-if analysis, and improve decision-making. Fig. 6 depicts a series of higher order Voronoi diagrams of $P = \{p_3, \ldots, p_8\}$ including $\mathcal{V}^{(1)}(P), \mathcal{V}^{(2)}(P), \mathcal{V}^{(3)}(P), \mathcal{V}^{(4)}(P), \mathcal{V}^{<2>}(P), \mathcal{V}^{<3>}(P), \mathcal{V}^{[4]}(P)$, and $\mathcal{V}^{[5]}(P)$. Emergency management officials can compare and contrast these diagrams to make effective and prompt decisions in highly dynamic environments. In addition, they can interactively insert/remove generators and analyze what-if situations as illustrated in Fig. 7 where a new generator p_9 is added to P.

4. Summary and future work

Disasters and emergency situations can lead to various forms of financial, structural and environmental damage. Even though it is almost impossible to avoid occurrences of disasters, effective prediction and preparedness along with an post emergency management program can mitigate the risk and damage (Jayaraman, Chandrasekhar, & Rao, 1997). What-if ordered and unordered k-nearest point queries, kth-neighboring queries, districting, facility locating queries and routing queries are therefore of great importance for emergency management as current GIS lack a dynamic data structure systematically supporting all these types of queries in highly active environments.

In this paper, we introduce a flexible Delaunay triangle based data structure and relevant algorithms supporting the complete $\mathcal{V}^{(k)}$, $\mathcal{V}^{<k>}$, and $\mathcal{V}^{[k]}$. We show how the data structure supports ordered/unordered k-nearest point queries, k-nearest zone queries, districting queries, facility locating queries and routing queries, and demonstrate how these Voronoi tessellations could be used in each phase of emergency management.

Future work will incorporate diverse generalized Voronoi diagrams and will consider background geospatial information

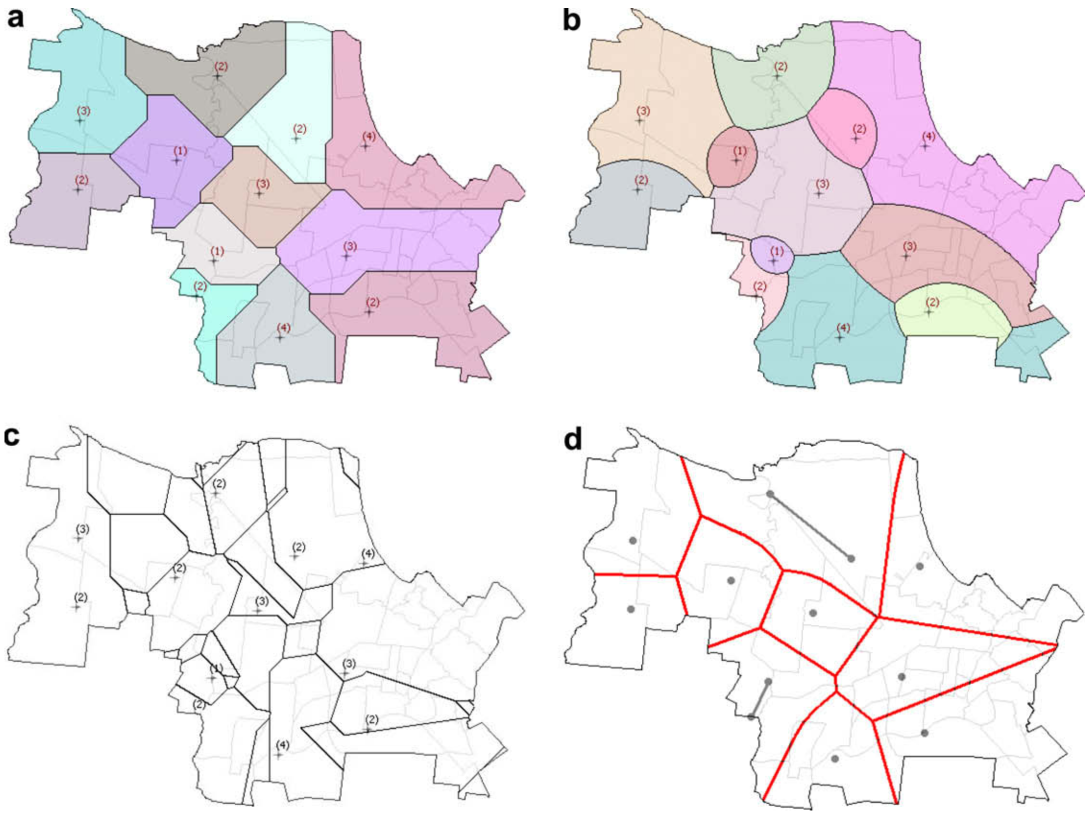

Fig. 8. Generalized Voronoi diagrams (P same as in Fig. 4) where numbers in brackets are weights: (a) the Manhattan Voronoi diagram of P; (b) the multiplicatively weighted Voronoi diagram of P; (c) the multiplicatively weighted Manhattan-metric order-2 Voronoi diagram P; (d) the line Voronoi diagram of P.

including population-at-risk and physical constraints. Fig. 8 depicts some extensions with the same dataset used in Fig. 4. Fig. 8a shows the Manhattan Voronoi diagram which is of particular interest in urban analysis where the Euclidean distance is not applicable. Fig. 8b depicts the multiplicatively weighted Voronoi diagram of the same dataset which is useful when objects convey different weights. Fig. 8c shows the combination of the first two with the same weight used in Fig. 8b. In addition, Fig. 8d shows the line Voronoi diagram where points belonging to the same suburb form a line segment. Even though these generalized Voronoi diagrams provide useful information for emergency management, all these require different data structures. Little research has been conducted on combining these data structures to provide a unified data structure for what-if analysis. We plan to incorporate these generalized Voronoi diagrams into the unified data structure providing a richer set of spatial tessellations.

References

Aurenhammer, F., & Schwarzkopf, O. (1991). A simple on-line randomized incremental algorithm for computing higher order Voronoi diagrams. In *Proceedings of the seventh annual symposium on computational geometry* (pp. 142–151). North Conway, NH: ACM Press.

Chang, N.-B., Wei, Y. L., Tseng, C. C., & Kao, C.-Y. J. (1997). The design of a GIS-based decision support system for chemical emergency preparedness and response in an Urban environment. *Computers, Environment and Urban Systems, 21*(1), 67–94.

Chazelle, B., & Edelsbrunner, H. (1987). An improved algorithm for constructing kth-order Voronoi diagrams. *IEEE Transactions on Computers, 36*(11), 1349–1354.

de Berg, M. T., van Kreveld, M. J., Overmars, M. H., & Schwarzkopf, O. (2000). *Computational geometry: Algorithms and applications* (2nd ed.). Heidelberg: Springer-Verlag.

Dehne, F. K. H. A. (1983). An $O(N^4)$ algorithm to construct all Voronoi diagrams for k-nearest neighbor searching in the euclidean plane. In J. Díaz (Ed.). *Proceedings of the international colloquium on automata, languages and programming, lecture notes in computer science* (Vol. 154, pp. 160–172). Barcelona, Spain: Springer.

Dymon, U. J., & Winter, N. L. (1993). Evacuation mapping: The utility of guidelines. *Disasters, 17*(1), 12–24.

Gahegan, M., & Lee, I. (2000). Data structures and algorithms to support interactive spatial analysis using dynamic Voronoi diagrams. *Computers, Environments and Urban Systems, 24*(6), 509–537.

Goodchild, M. F. (2006). GIS and disasters: Planning for catastrophe. *Computers, Environment and Urban Systems, 30*(3), 227–229.

Haddow, G. D., & Bullock, J. A. (2003). *Introduction to emergency management*. Butterworth-Heinemann.

Hanjoul, P., Beguin, H., & Thill, J.-C. (1989). Advances in the theory of market areas. *Geographical Analysis, 21*(3), 185–196.

Jayaraman, V., Chandrasekhar, M., & Rao, U. (1997). Managing the natural disasters from space technology inputs. *Acta Astronautica, 40*, 291–325.

Johnson, R. (1994). GIS technology for disasters and emergency management. Technical Report J-8474, ESRI. <http://www.esri.com/library/whitepapers/pdfs/disastermgmt.pdf>.

Kevany, M. J. (2003). GIS in the world trade center attack – trial by fire. *Computers, Environment and Urban Systems, 27*(6), 571–583.

Lee, D.-T. (1982). On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Transactions on Computers, 31*(6), 478–487.

Lee, I., & Gahegan, M. (2002). Interactive analysis using Voronoi diagrams: Algorithms to support dynamic update from a generic triangle-based data structure. *Transactions in GIS, 6*(2), 89–114.

Mehlhorn, K., & Näher, S. (1999). *LEDA a platform for combinatorial and geometric computing*. Cambridge, UK: Cambridge University Press.

Montoya, L. (2003). Geo-data acquisition through mobile gis and digital video: An Urban disaster management perspective. *Environmental Modelling & Software, 18*(10), 869–876.

Okabe, A., Boots, B. N., & Sugihara, K. (1994). Nearest neighbourhood operations with generalized Voronoi diagrams: A review. *International Journal of Geographical Information Systems, 8*(1), 43–71.

Okabe, A., Boots, B. N., Sugihara, K., & Chiu, S. N. (2000). *Spatial tessellations: Concepts and applications of Voronoi diagrams* (2nd ed.). West Sussex: John Wiley & Sons.

Rushton, G., & Thill, J.-C. (1989). The effect of distance metric on the degree of spatial competition between firms. *Environment and Planning A, 21*, 499–507.

Salt, C. A., & Dunsmore, M. C. (2000). Development of a spatial decision support system for post-emergency management of radioactively contaminated land. *Journal of Environmental Management, 58*(3), 169–178.

Samet, H. (2006). *Foundations of multidimensional and metric data structures*. San Francisco: Morgan Kaufman.

Shamos, M. I., & Hoey, D. (1975). Closest-Point Problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science* (pp. 151–162). IEEE Computer Society.

Thill, J.-C., & Rushton, G. (1992). Demand sensitivity to space-price competition with manhattan and euclidean representations of distance. *Annals of Operations Research, 40*, 381–401.

Wisner, B., Blaikie, P., Cannon, T., & Davis, I. (2003). *At risk: Natural hazards, people's vulnerability and disasters* (2nd ed.). Routledge.

Zerger, A., & Smith, D. I. (2003). Impediments to using GIS for real-time disaster decision support. *Computers, Environment and Urban Systems, 27*(2), 123–141.