Paper presented at: Sixth annual International Symposium on Voronoi Diagrams in science and engineering (ISVD), Copenhagen, 2009, June 23-26

Capacity-Constrained Voronoi Diagrams in Continuous Spaces

Michael Balzer
University of Konstanz, Germany
michael.balzer@gmail.com

***Abstract*—A Voronoi diagram of a set of sites partitions a bounded space into regions of different areas. A capacity-constrained Voronoi diagram is a partition in which the area for each Voronoi region is predefined. In this paper, we present two approaches for computing such capacity-constrained Voronoi diagrams in continuous spaces. Our first approach is based on ordinary (non-weighted) distance functions and achieves the capacity constraint by a general optimization of the site locations. Our second approach is based on weighted distance functions and optimizes the weights of the sites. Both approaches are iterative methods that start with an initial set of sites and then optimize the area of one Voronoi region at a time. As a consequence, the dimensionality of the individual optimization problem is minimal, which enables a reliable and fast convergence even for large sets of sites.**

I. Introduction

A Voronoi diagram of a set of n sites partitions the Euclidean space into n regions. The shapes of the regions are depending on the neighbor topology according to the Delaunay graph. The region areas are implicitly determined by the distances between the neighboring sites. By additionally assigning a set of n weights to the set of sites, and varying the distance function to incorporate these weights, it is possible to influence the shape and especially the area of the Voronoi regions. However, the actual areas of the regions are still implicitly determined by the neighbor topology and distances, and their exists no direct relation between the locations and/or weights of the sites and the areas of the resulting Voronoi regions.

In this paper, we are interested in Voronoi diagrams in which the region areas are given as a constraint. This means that the resulting areas for all regions are predefined, and the locations and/or weights have to be determined in such a way that these constraints are fulfilled. We call this predefined region area the *capacity* of the site, and a Voronoi diagram in which each regions has an area that is equal to the capacity of the respective site a *capacity-constrained Voronoi diagram*. The motivation for these diagrams is given by applications in computer graphics [1] and information visualization [2].

Generating such Voronoi diagrams with capacity constraints is an optimization problem of high dimensionality. For ordinary (non-weighted) distance functions, the dimensionality of the optimization problem is nd for n sites in d-dimensional space, where the site locations are the parameters that have to be optimized. For weighted distance functions, the dimensionality is n with the site weights as the optimized parameters.

Current algorithms for generating capacity-constrained Voronoi diagrams are restricted to specific distance functions, such as Aurenhammer et al. [3], Ohyama [4], and Balzer and Heck [5] for power diagrams, or Reitsma et al. [6] for multiplicatively weighted Voronoi diagrams. The latter two are also restricted to discrete spaces with the drawbacks of limited accuracy and missing geometric bisector representation.

In the following sections, we present two approaches for computing capacity-constrained Voronoi diagrams in continuous spaces. The first approach is based on ordinary distance functions and optimizes the site locations to fulfill the capacity constraint. The second approach is based on weighted distance functions and optimizes the weights of the sites. Both approaches are iterative methods which optimize one Voronoi region at a time. As a consequence, the dimensionality of the individual optimization problem is minimal. This enables a reliable and fast convergence of the optimization even for large sets of sites, especially for our second approach which uses weighted distance functions. Further note that our approaches are not restricted to specific distance functions and can therefore be utilized for a variety of different Voronoi diagrams.

The remainder of the paper is structured as follows: Section II discusses the necessary theoretic background. Section III presents our two approaches for the computation of capacity-constrained Voronoi diagrams in continuous spaces. Finally, a short conclusion is given in Section IV.

II. Background

In this section, we provide the necessary theoretical background to the aspects of Voronoi diagrams that are considered in this paper. For the ease of exposition, we restrict ourselves to the Euclidean plane, without loss of generality. For a comprehensive overview of Voronoi diagrams and their properties see [7], [8].

A. Ordinary Voronoi Diagrams

We consider a set $S = \{s_1, \ldots, s_n\}$ of n points in the Euclidean plane $\mathbb{R}^2$, and assume that $2 \leq n < \infty$. The n points have the Cartesian coordinates $(s_{11}, s_{12}), \ldots, (s_{n1}, s_{n2})$ and are distinct in the sense that $(s_{i1}, s_{i2}) \neq (s_{j1}, s_{j2})$ for $i \neq j$, $i, j \in I_n = \{1, \ldots, n\}$. These points in the set S are the *sites*.

Let x be an arbitrary point in $\mathbb{R}^2$ with coordinates (x_1, x_2). The Euclidean distance between the point x and a site $s_i \in S$ is given by

$$d_E(s_i, x) = \|s_i - x\| = \sqrt{(s_{i1} - x_1)^2 + (s_{i2} - x_2)^2}. \quad (1)$$

If $s_i \in S$ is the nearest site from x or one of the nearest sites from x, we have the relation $d_E(s_i, x) \leq d_E(s_j, x)$ for $i \neq j$, $j \in I_n$. We call the points with the nearest site s_i, given by

$$V_E(s_i) = \{x \mid d_E(s_i, x) \leq d_E(s_j, x), i \neq j, j \in I_n\}, \quad (2)$$

the *ordinary Voronoi region* associated with s_i, and the set given by

$$\mathscr{V}_E(S) = \{V_E(s_1), \ldots, V_E(s_n)\} \quad (3)$$

the *ordinary Voronoi diagram* generated by S. The *bisector* of two regions $V_E(s_i)$ and $V_E(s_j)$, with $i \neq j$, is the line perpendicularly to the line segment $\overline{s_i s_j}$, formed by points equidistant to both s_i and s_j, and divides the plane in two half-planes.

B. Weighted Voronoi Diagrams

In an ordinary Voronoi diagram of a set S of n sites it is implicitly assumed that the sites are identical except for their locations, or that each site has the same weight. As an extension, a set of parameters $W = \{w_i \mid i \in \{1, \ldots, n\}\}$ may be given. These parameters are the *weights*. By using weighted sites, we can define *weighted distances* that are used for generating *weighted Voronoi diagrams* $\mathscr{V}(S, W)$. The Voronoi regions $V(s_i, w_i) \in \mathscr{V}(S, W)$ not only depend on the locations of the sites, but also on their weights, where usually sites with larger weights form larger Voronoi regions. Since the weighted distance allows many functional forms, a wide variety of weighted Voronoi tessellations exist, all having their own characteristics. In this paper, we consider two types of weighted distance functions that are especially suitable for computing capacity-constrained Voronoi diagrams due to their connected regions.

1) Additively Weighted Voronoi Diagrams: An *additively weighted Voronoi diagram* $\mathscr{V}_{AW}(S, W)$ is characterized by the following *AW distance function* between a site $s_i \in S$ with its assigned weight $w_i \in W$ and a point x:

$$d_{AW}(s_i, w_i, x) = \|s_i - x\| - w_i. \quad (4)$$

The shape of the bisector between two sites s_i and s_j varies according to the parameter values $\alpha = \|s_i - s_j\|$ and $\beta = w_i - w_j$, where $\beta \geq 0$ is assumed without loss of generality. If $0 < \alpha < \beta$, the site s_i dominates the whole plane, and the region of s_j disappears. If $\alpha = \beta$, the region of s_j disappears except for the half-line radiating from s_j in the direction from s_i to s_j. Finally, if $\alpha > \beta$, the bisector of $V_{AW}(s_i, w_i)$ and $V_{AW}(s_j, w_j)$ forms a hyperbolic curve with foci s_i and s_j. Note that if $\beta = 0$, the bisector again becomes a straight line perpendicularly bisecting the line segment $\overline{s_i s_j}$ through its midpoint.

2) Power Diagrams: A *power diagram* $\mathscr{V}_{PW}(S, W)$, which is also known as an *additively weighted power Voronoi diagram*, is characterized by the following *PW distance function* between a site $s_i \in S$ with its assigned weight $w_i \in W$ and a point x:

$$d_{PW}(s_i, w_i, x) = \|s_i - x\|^2 - w_i. \quad (5)$$

The bisector of two Voronoi regions $V_{PW}(s_i, w_i)$ and $V_{PW}(s_j, w_j)$ is a straight line. It corresponds to the line perpendicularly bisecting the line segment $\overline{s_i s_j}$ through the point p^*_{ij} given by

$$p^*_{ij} = \frac{\|s_j\|^2 - \|s_i\|^2 + w_i - w_j}{2\|s_j - s_i\|^2}(s_j - s_i). \quad (6)$$

If $\|s_i - s_j\|^2 < w_j - w_i$, then the site s_i does not reside within its Voronoi region $V_{PW}(s_i, w_i)$.

C. Centroidal Voronoi Diagrams

A *centroidal Voronoi diagram* is a Voronoi diagram in a bounded space $\Omega \subset \mathbb{R}^2$ with the property that each site s_i coincides with the centroid of its Voronoi region $V(s_i)$. The *centroid* m_i of a Voronoi region $V(s_i)$ is calculated by

$$m_i = \frac{\int_{V(s_i)} x\rho(x)dx}{\int_{V(s_i)} \rho(x)dx}, \quad (7)$$

where $\rho(x) \geq 0$ is a given density function defined in Ω. In other words, a centroid is the center of mass of a Voronoi region with respect to the density function. The importance of centroidal Voronoi diagram is founded on its relationship with the energy function

$$\mathscr{F}(S, \mathscr{V}(S)) = \sum_{i=1}^{n} \int_{x \in V(s_i)} \rho(x)\|x - s_i\|^2 dx, \quad (8)$$

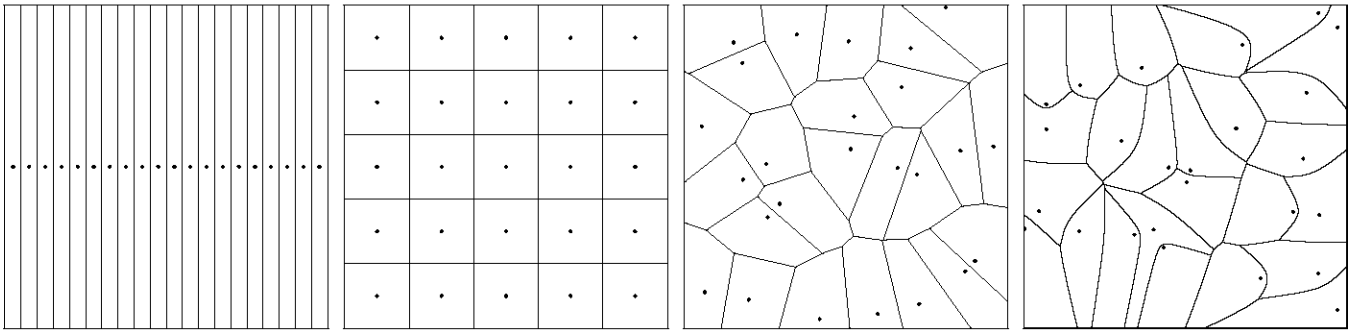
where $V(s_i) \in \mathscr{V}(S)$ and $s_i \in S$. A necessary condition for $\mathscr{F}$ to be minimized is that $\mathscr{V}(S)$ is a centroidal Voronoi diagram of S.

A common approach to generate a centroidal Voronoi diagram is to employ Lloyd's method [9]. This algorithm iteratively moves each site $s_i \in S$ to the centroid m_i of the corresponding Voronoi region $V(s_i) \in \mathscr{V}(S)$ until the sites meet some convergence criterion. Due to the fact that each relocation of a site to its centroid reduces the energy in Equation 8, the algorithm converges to a local minimum of $\mathscr{F}$, in which each site coincides with the centroid of its corresponding Voronoi region. For a comprehensive treatment of the topic, we refer to [10].

D. Capacity-Constrained Voronoi Diagrams

Consider a set S of n sites that determines a Voronoi diagram $\mathscr{V}(S)$ in the space $\Omega \subseteq \mathbb{R}^2$ with the density function $\rho(x) \geq 0, x \in \Omega$. The area of the Voronoi region $V(s_i) \in \mathscr{V}(S)$ of a site $s_i \in S$ is given by

$$|V(s_i)| = \int_{x \in V(s_i)} \rho(x)dx. \quad (9)$$



(a) collinear sites (b) sites arranged in a regular grid (c) non-regularly distributed sites using the ordinary Euclidean distance (d) non-regularly distributed sites using the weighted AW distance

Figure 1. Examples of capacity-constrained Voronoi diagrams for 25 sites. Each Voronoi region in these diagrams has the same capacity.

This Voronoi region area $|V(s_i)|$ is denoted as the *capacity* of s_i.

A set of parameters $C = \{c_i \mid i \in \{1,\ldots,n\}\}$ with $0 < c_i \leq \infty$ and $\sum C = \int_{x\in\Omega} \rho(x)dx$ may be given, which is called the *capacity constraint*. A *capacity-constrained Voronoi diagram* $\mathscr{V}(S,C)$ is a Voronoi diagram of S in Ω in which each site $s_i \in S$ has the corresponding capacity $c_i \in C$. In other words, a capacity-constrained Voronoi diagram $\mathscr{V}(S,C)$ with n sites $s_i \in S$ and a capacity constraint C with n parameters fulfills the condition

$$\sum_{i=1}^{n} \left(|V(s_i)| - c_i\right)^2 = 0. \tag{10}$$

The space Ω can be bounded or unbounded as long as the area of that space combined with the density function ρ is equal to the sum of capacities in C. In this paper, we utilize only bounded spaces without loss of generality, for which each site has a finite capacity, to allow a clearer presentation of the algorithms.

The concept of capacity-constrained Voronoi diagrams is focusing on the resulting Voronoi regions rather then on the underlying site locations. Its intention is to determine partitions of given spaces that consist of compact regions of predefined size. It is thereby not restricted to particular spaces or distance functions.

Trivial examples for capacity-constrained Voronoi diagrams are sets of collinear sites as in Figure 1(a) or regular lattices of sites as in Figure 1(b), both with equal capacities for all sites. More relevant in practical applications are cases with non-regular site distributions as shown in Figure 1(c) and Figure 1(d), or with capacity constraints where each Voronoi region has a different, individually defined, area.

III. Computation

The shapes and areas of the regions in a Voronoi diagram are implicitly given by the neighbor topology and the pairwise distances between these neighbors. Sites with no nearby neighbors form larger regions than sites that are closely surrounded by their neighbors. Weighted distance functions allow to increase the area of a region by increasing the weight of the respective site. Unfortunately, due to the interdependency of the sites, there exists no direct relation between the locations and/or weights of the sites and the resulting Voronoi region areas. To generate a capacity-constrained Voronoi diagram it is therefore necessary to perform an iterative optimization of the site parameters based on the difference between the actual areas of the Voronoi regions and the given capacity constraint.

In this section, we present two approaches for computing capacity-constrained Voronoi diagrams in continuous spaces. Both methods are iterative algorithms that improve the region area of one single site at a time, and converge towards a stable state in which the capacity constraint is fulfilled. The first approach is based on ordinary (non-weighted) distance functions and modifies the locations of the sites. In contrast, the second approach uses weighted distance functions and modifies the weights of the sites.

A. Using Ordinary Distance Functions

The Voronoi region $V(s_i)$ of a site $s_i \in S$ in an ordinary Voronoi diagram $\mathscr{V}(S)$ is solely determined by the relative location of all sites $s_j \in S$ that are neighbors to s_i. The generation of a capacity-constrained ordinary Voronoi diagram $\mathscr{V}(S,C)$ for n sites with their capacities in C in the d-dimensional space $\Omega \subseteq \mathbb{R}^d$ can be formulated as an optimization problem in nd dimensions. The variables in this optimization are the d-dimensional locations of all n sites. The continuous function $\mathscr{D}$ that has to be minimized is the total difference between the capacity constraint in C and the actual areas of the Voronoi regions:

$$\mathscr{D}\left(\mathscr{V}(S), C\right) = \sum_{i=1}^{n} \left(|V(s_i)| - c_i\right)^2 \text{ with } s_i \in S, c_i \in C. \tag{11}$$

This function is equivalent to Equation 10. The global minimum $\mathscr{D} = 0$ is achieved if the Voronoi diagram $\mathscr{V}(S)$

fulfills the given capacity constraint C. Such global minimum always exists, which is obvious if one considers a collinear set of n sites for which a solution can easily be found. Actually, there exists not only one set of sites in S that fulfills the capacity constraint in C, but rather a large number of variants for which $\mathscr{D}$ attains the global minimum.

The minimization of the continuous function $\mathscr{D}$ can be performed with general optimization techniques that use the site locations as the parameter vector of dimensionality nd. The drawback of such direct minimization approach is that for large sets of sites the high dimensionality of the parameter vector entails a prohibitive runtime and numerical problems during the optimization. This problem of the high dimensionality of the optimization can be remedied by not optimizing all site locations at once, but rather iteratively optimizing just one site location at a time. This reduces the dimensionality of an optimization step from nd to d, where d—the dimension of the space Ω—is usually small, and does not depend on the number of sites. Ultimately, this iterative improvement of single site locations is equivalent to the optimization of all sites at once. The only difference is that the iterative approach subdivides the full optimization into smaller fragments, which are easier to solve. The result is the following approach for the computation of a capacity-constrained ordinary Voronoi diagram, which is outlined in Algorithm 1.

The input for the algorithm is a set S of n sites, a set C of n associated capacities, and a d-dimensional space Ω. The result of the algorithm is a capacity-constrained ordinary Voronoi diagram $\mathscr{V}(S,C)$ in Ω with $\mathscr{D}(\mathscr{V}(S),C) = 0$. Initially, the Voronoi diagram $\mathscr{V}(S)$ is computed with the given sites in S. This diagram is then iteratively optimized towards $\mathscr{D} = 0$ by performing the following steps in each iteration:

1) A set $I := \{1,\ldots,n\}$ is initialized, which represents the indices of the sites in S that have not been optimized in this iteration.
2) While the set I is not empty, which means that there are sites that have not been optimized in this iteration:
   a) Choose the site s_i with index $i \in I$ for which $|V(s_i)| - c_i$ is minimal for all the sites that have not been optimized in this iteration. The site s_i is thereby the smallest non-optimized site in this iteration with respect to its intended capacity c_i.
   b) Minimize $\mathscr{D}(\mathscr{V}(S),C)$ by adjusting the location of s_i and simultaneously recomputing the Voronoi diagram $\mathscr{V}(S)$.
   c) Remove index i from the set I, which indicates that s_i has been optimized in this iteration.
3) If none of the site locations has changed during this iteration, then the optimization converged to a stable state, and the algorithm terminates.

The priority for the sites implemented in step 2(a) and line 6 in Algorithm 1 significantly improves the convergence of the algorithm. The reason for this is that a site with an undersized Voronoi region just has to identify the most oversized and/or least undersized neighboring site, and then directly moves into this direction thereby increasing its own area and decreasing the other's area. In contrast, a site with an oversized region must avoid other sites to decrease its area. Rather, oversized regions have to wait until undersized regions indirectly decrease them. Hence, prioritizing the sites with undersized regions effectively moves them towards oversized regions which profit thereof later in the iteration.

Algorithm 1: Using Ordinary Distance Functions

Input: Sites $S = \{s_1,\ldots,s_n\}$,
Capacity constraint $C = \{c_1,\ldots,c_n\}$,
d-dimensional space Ω with $\sum C = \int_{x\in\Omega} dx$

Output: Capacity-constrained ordinary Voronoi diagram $\mathscr{V}(S,C)$

```
1  Compute the Voronoi diagram V(S);
2  repeat
3      stable := true;
4      Initialize a set I := {1,...,n};
5      while I ≠ ∅ do
6          Choose the site s_i with i ∈ I, s_i ∈ S and
           |V(s_i)| − c_i < |V(s_j)| − c_j for each
           j ∈ I, j ≠ i, s_j ∈ S;
7          Minimize D(V(S),C) by adjusting the
           location of s_i and recomputing the Voronoi
           diagram V(S);
8          if the location of s_i changed then
9              stable := false;
10         Remove i from I;
11 until stable = true ;
```

The minimization in step 2(b) and line 7 in Algorithm 1 can be performed by any optimization algorithm that works on function samples. A good choice is the Downhill simplex method [11] because it reliably minimizes the function locally around a given starting location, which is preferably the current site location. Furthermore, it is not necessary that the minimization for each individual site is performed until it finds a local minimum. Rather, a few minimization steps are sufficient, which results in a much shorter runtime than performing a complete minimization.

Due to the fact that the given space Ω may be bounded, it is possible that some resulting site locations are not in Ω if the minimization is solely based on Equation 11. This can be avoided by adding a penalty to site locations that are not in Ω. Good results were generated by using a penalty function $\mathscr{P}_1$ that is applied to the currently optimized site

s_i and then added to the value of $\mathscr{D}$ with

$$\mathscr{P}_1(s_i) = \begin{cases} 0 & \text{if } s_i \in \Omega, \\ (\|s_i - \Omega\| c_i)^2 & \text{if } s_i \notin \Omega \end{cases}. \tag{12}$$

This function adds a capacity-dependent penalty to $\mathscr{D}$ based on the distance of the site location to Ω.

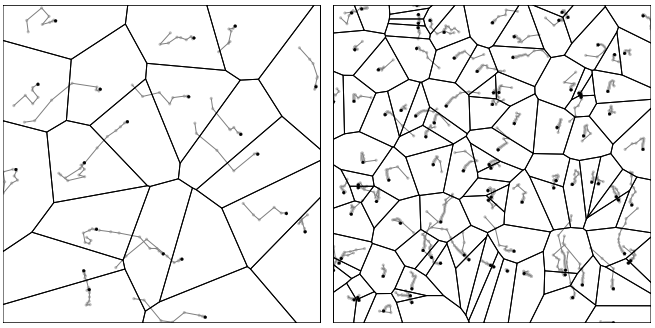
Another effect of the presented algorithm is that sometimes two or more neighboring sites are moved very close together, almost having the same site location. This occurs if neighboring sites possess regions that are much larger then their desired capacity. Then the minimization moves these sites closer together, and as result, their combined region area is reduced. Even though such Voronoi diagrams are still valid and fulfill the capacity constraint, the resulting close sites are often unfavorable with regard to the intended application. This problem is remedied by applying another penalty $\mathscr{P}_2$ to the currently optimized site s_i that is then added to the value of $\mathscr{D}$ with

$$\mathscr{P}_2(s_i) = \sum_{j=1}^{n} \begin{cases} 0 & \text{if } i = j \text{ or } \delta >= 1, \\ (1-\delta)\, c_i c_j & \text{if } i \neq j \text{ and } \delta < 1, \end{cases} \quad \text{with } \delta = \frac{\|s_i - s_j\|}{\sqrt{c_i + c_j}} \tag{13}$$

This function adds a capacity-dependent penalty if the site is very close to another site, and no penalty if the site does not have nearby neighboring sites.

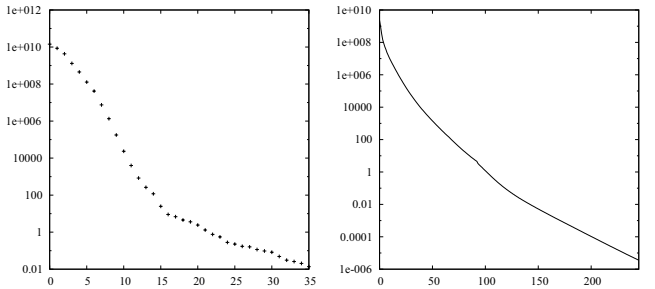
The convergence of the minimization can be guaranteed if the minimization algorithm in step 2(b) and line 7 does not increase the value of $\mathscr{D}$. It is not guaranteed that the algorithm terminates in a global minimum. However, during extensive tests, the algorithm always terminated in the global minimum $\mathscr{D} = 0$, and never in a local minimum $\mathscr{D} > 0$. If it happens that a local minimum with $\mathscr{D} > 0$ is achieved, it should be sufficient to move all sites that do not fulfill their capacity constraint to new random locations within Ω, and then to proceed with the minimization. If these potential cases of local minima are neglected, the algorithm shows clear logarithmic convergence towards $\mathscr{D} = 0$. This convergence is preserved if one or both of the penalty functions $\mathscr{P}_1$ and $\mathscr{P}_2$ are utilized.

Two examples for capacity-constrained ordinary Voronoi diagrams generated with Algorithm 1 are given in Figure 2. The diagram in example (a) consists of 20 sites with equal capacities, whereas the diagram in example (b) consists of 100 sites with different capacities. The computation of example (a) is further illustrated in Figure 10 at the end of the paper by a sequence of iterations. The development of the capacity error $\mathscr{D}$ of these two examples is given in Figure 3, showing a clear logarithmic convergence. The computation of the example with 20 sites required 35 iterations in less than 5 seconds on Intel Core 2 hardware, computing more than 21000 function samples of $\mathscr{D}$ for the minimization in step 2(b), including the update of the Voronoi diagram for each sample and the extraction of the individual polygons. The computation time for the necessary 245 iterations for the 100 sites example was approximately 16 minutes, computing more than 3.2 million function samples.



(a) 20 sites with equal capacities (b) 100 sites with different capacities

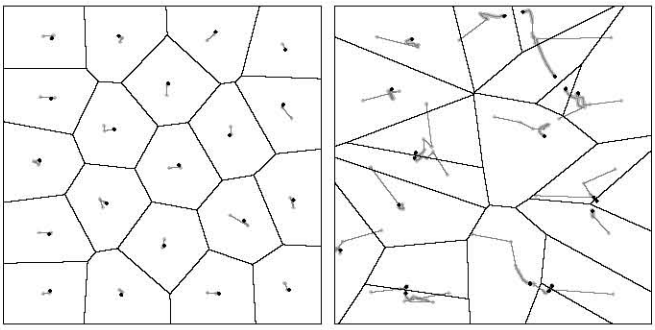
Figure 2. Capacity-constrained ordinary Voronoi diagrams computed with Algorithm 1. The traces illustrate the site movements during the computation.



(a) 20 sites with equal capacities (b) 100 sites with different capacities

Figure 3. Development of the function $\mathscr{D}$ for each iteration of the examples in Figure 2.

The convergence and the result of an individual computation depends on the initial distribution of sites. We can use this to our advantage if all sites have similar capacities. In this case, we can at first generate an approximation of a centroidal Voronoi diagram with a few iterations of Lloyd's method, and then use the resulting sites as the initial set for the computation of a capacity-constrained ordinary Voronoi diagram. The result of this combination is usually a homogeneous distribution of sites that all reside near the centroids of their corresponding Voronoi regions. Additionally, the convergence of the algorithm is greatly improved by this variant due to the fact that a centroidal Voronoi diagram is a good first approximation of a capacity-constrained ordinary Voronoi diagram with equal capacities. An example for the development of a Voronoi diagram with this variant of our algorithm is given in Figure 4(a). Unfortunately, the same effect cannot be observed for sets of sites with differing capacities because of their dissimilarity to centroidal Voronoi diagrams.



(a) 20 sites with equal capacities (b) 20 sites with different capacities

Figure 4. Capacity-constrained ordinary Voronoi diagrams with 20 sites that used centroidal Voronoi diagrams as initial sets of sites for the computation.

The values for the computation time of the presented examples show that the computation of a capacity-constrained ordinary Voronoi diagrams is very expensive. Especially large sets of sites, with many thousands or even millions of sites are beyond computational feasibility. The main reason for this is that each function sample during the minimization consists of the update of one site in the Voronoi diagram with $O(\log n)$ and the evaluation of the area of each Voronoi region with $O(n)$. Hence, the resulting computational time complexity for one single function sample is $O(\log n + n)$. The time complexity for a complete iteration is $O(n^2 + n \log n)$.

B. Using Weighted Distance Functions

The Voronoi region $V(s_i, w_i)$ of a site $s_i \in S$ in a weighted Voronoi diagram $\mathscr{V}(S,W)$ is determined by the relative location of all sites $s_j \in S$ that are neighbors to s_i. The distance function that determines the neighbor topology and forms the bisectors additionally incorporates a weight $w_i \in W$ for each site $s_i \in S$. Thus, the set W of weights offers an additional degree of freedom to control the resulting Voronoi regions.

The incorporation of a weight parameter in the distance computation for a weighted Voronoi diagram $\mathscr{V}(S,W)$ can have many functional forms. The common distance functions either use an *additive* or a *multiplicative* weight term. Instances for the first type are the AW and PW distance functions presented in Section II. An example for the latter type is the multiplicatively weighted Voronoi diagram that uses the distance function $d_{MW}(s_i, w_i, x) = \frac{1}{w_i}\|s_i - x\|, w_i > 0$. Other weighted Voronoi diagrams can either be reduced to these two types, or at least possess similar characteristics. At the bottom line, all weighted distance functions allow to increase or decrease the area of a Voronoi region by solely increasing and/or decreasing the weight of the respective site. The modification of the site locations is typically not necessary to achieve a given capacity constraint [3], [5]. The advantage of distance functions with additive weights is that they result in connected Voronoi regions, whereas multiplicative weight terms often generate disconnected regions.

The generation of a capacity-constrained weighted Voronoi diagram $\mathscr{V}(S,W,C)$ for n sites with their respective weights in W and their capacities in C in the d-dimensional space $\Omega \subset \mathbb{R}^d$ can be formulated as an optimization problem in n dimensions. The variables in this optimization are the weights of all n sites. The continuous function $\mathscr{D}(\mathscr{V}(S,W),C)$ that has to be minimized is equivalent to the function for the ordinary case in Equation 11. A weighted Voronoi diagram $\mathscr{V}(S,W)$ that fulfills the given capacity constraint C is achieved if $\mathscr{D} = 0$, representing the global minimum of $\mathscr{D}$. Similar to ordinary distance functions, such global minimum can be determined by general optimization methods that work on function samples. Here, the same problem occurs as for ordinary distance functions: For large sets of sites the high dimensionality of the parameter vector entails a prohibitive runtime and numerical problems during the optimization.

The solution to this problem of high dimensionality for the optimization can again be remedied by not optimizing all site weights at once, but rather iteratively optimizing just one site weight at a time. In contrast to the algorithm for ordinary distance functions, we optimize the following function $\mathscr{D}'$ during each minimization of one single site:

$$\mathscr{D}'(\mathscr{V}(S,W), C, i) = (|V(s_i, w_i)| - c_i)^2. \tag{14}$$

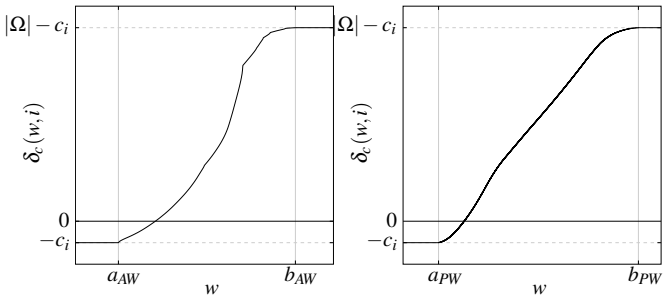
This reduces the dimensionality of an optimization step from n to 1.

The minimization of $\mathscr{D}'$ for a single site $s_i \in S$ is performed by adjusting the weight $w_i \in W$. For analyzing the correlation between a weight w that is associated with a site s_i, and the area of the resulting Voronoi region $V(s_i, w) \in \mathscr{V}(S,W)$ in a bounded continuous space Ω, we consider the capacity error

$$\delta_c(w, i) = |V(s_i, w)| - c_i. \tag{15}$$

It becomes apparent that this capacity error δ_c is a continuous function with three characteristic intervals, illustrated in Figure 5. In the first interval $w \in (-\infty, a]$, the site s_i is dominated by the other sites $s_j \in S, i \neq j$, thus the Voronoi region $V(s_i, w)$ disappears. The capacity error in this interval is constant with $\delta_c(w, i) = -c_i$. In the second interval $w \in (a, b)$, the Voronoi region of the site s_i occupies some part but not the whole area of Ω. In this interval, $\delta_c(w, i)$ is monotonically increasing, and passing its root $\delta_c(w, i) = 0$. In the third interval $w \in [b, \infty)$, the site s_i dominates the whole bounded space Ω, having a constant capacity error of $\delta_c(w, i) = |\Omega| - c_i$.

This behavior of the capacity error is in general independent of the utilized distance function. The lower bound for the capacity error is always $\delta_c = -c_i$ and the upper



(a) AW distance function (b) PW distance function

Figure 5. Correlation between the weight w of a site s_i and the resulting capacity error $\delta_c(w,i)$. The weights of all other sites $s_j \in S, i \neq j$ remained fixed.

bound is always $\delta_c = |\Omega| - c_i$. Of course, the actual weights a and b that determine the intervals highly depend on the utilized distance function, and on the individual locations and weights of the other sites. For example, the plots in Figure 5 were generated in a square with edge length 1000 using a set of 10 random sites. The weights of all sites s_j other than s_i were $w_j = 0$. The actual interval bounds for the AW distance function were at $a_{AW} \approx -200$ and $b_{AW} \approx 626$, and for the PW distance function they were at $a_{PW} \approx -89860$ and $b_{PW} \approx 538067$.

The global minimum of $\mathscr{D}'$ is achieved if the capacity error for each site is zero, $\delta_c(w_i,i) = 0$. Hence, we have to find the root of the function δ_c for each site, which is simple due to its predictable behavior. A good method for finding the root of δ_c is the false position method, which reliably converges towards $\delta_c = 0$ at a faster rate than the also reliable bisection method. In contrast, methods that utilize the function's gradient, such as the Newton's method or the secant method, are not sufficient due to the constant values of $\delta_c(w,i)$ for $w \leq a$ and $w \geq b$ with respect to Figure 5. The result is the following approach for the computation of capacity-constrained weighted Voronoi diagrams, which is outlined in Algorithm 2.

The input for the algorithm is a set S of n sites, a set C of n associated capacities, and a d-dimensional space Ω. The result of the algorithm is a set W of n weights that determines a capacity-constrained weighted Voronoi diagram $\mathscr{V}(S,W,C)$ in Ω with $\mathscr{D}(\mathscr{V}(S,W),C) = 0$. The algorithm starts with the initialization of the set W of n weights with $w_i = 0$, $w_i \in W$. Then, the initial Voronoi diagram $\mathscr{V}(S,W)$ is iteratively optimized towards the capacity constraint C by adjusting the weights of the sites one at a time. The weight adjustment for each site s_i is performed by finding the root w of the function δ_c via the false position method. This root w is then assigned to the weight w_i. The overall algorithm stops if the weight of not even one site can be further improved.

The convergence of the algorithm is not guaranteed due to the fact that the optimization is based on the per site function

Algorithm 2: Using Weighted Distance Functions

Input: Sites $S = \{s_1, \ldots, s_n\}$,
Capacity constraint $C = \{c_1, \ldots, c_n\}$,
d-dimensional space Ω with $\sum C = \int_{x \in \Omega} dx$

Output: Capacity-constrained weighted Voronoi diagram $\mathscr{V}(S,W,C)$

```
1 Initialize a set of n weights W := {0,...,0};
2 repeat
3     stable := true;
4     foreach site s_i ∈ S do
5         Find the weight w such that δ_c(w,i) = 0 via the
          false position method;
6         if w_i ≠ w then
7             stable := false;
8         w_i := w;
9 until stable = true ;
```

$\mathscr{D}'$, and not on the global function $\mathscr{D}$. Thus, the improvement of the capacity error for one site may result in a higher deterioration of the capacity error of another site, which increases the value of $\mathscr{D}$. However, similar to the algorithm for ordinary distance functions, we could not observe cases where the algorithm did not converge to the global minimum $\mathscr{D} = 0$. Rather, the expansion and contraction of the regions always combined in a way that consistently reduced the overall capacity error towards the global minimum.

Four examples of capacity constrained weighted Voronoi diagrams generated with Algorithm 2 are presented in Figure 6. Two of them are based on 20 sites with equal capacities, and the other two are based on 100 sites with different capacities. The computation of the lower left example is further illustrated in Figure 11 at the end of the paper. In the AW distance function results, the elongated Voronoi regions emerge du to the fact that the sites always reside within their corresponding Voronoi regions. In contrast, the results using the PW distance function exhibit much more compact Voronoi regions, but the sites do not necessarily reside within their corresponding regions. The development of the function $\mathscr{D}$ that lead to these result is shown in the top row of Figure 8. These plots clearly illustrate the superlinear convergence of $\mathscr{D}$ throughout the computation using Algorithm 2. An overview of the computation time, the number of iterations, the number of necessary Voronoi diagram updates for evaluating δ_c during the root finding, and the minimum and maximum weights of the results are presented in Figure 9. Note, that the much larger computation time for the AW distance function examples is reasoned by the hyperbolic bisectors of the resulting regions. Most of the computation time was used to extract these hyperbolic

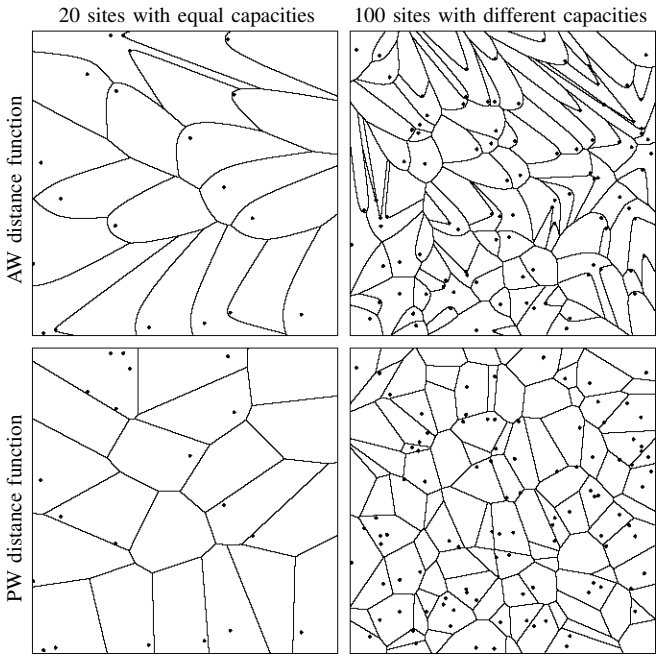


Figure 6. Capacity-constrained weighted Voronoi diagrams computed with Algorithm 2.

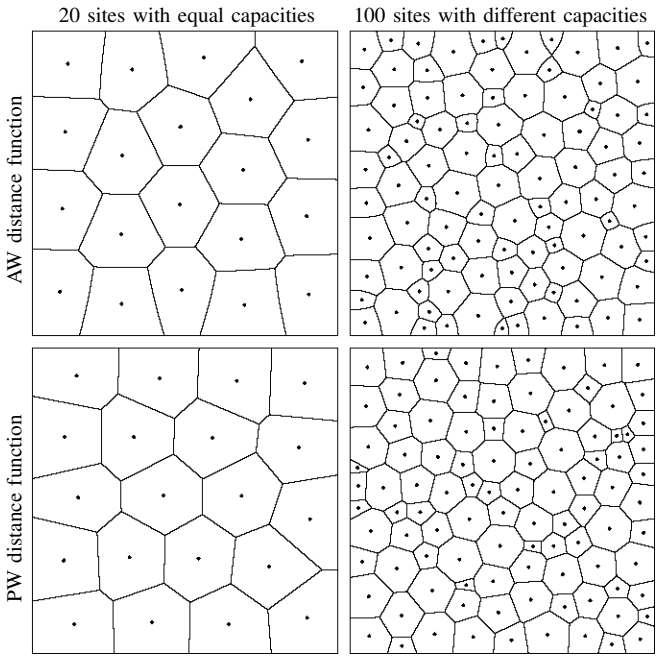


Figure 7. Centroidal capacity-constrained weighted Voronoi diagrams computed with the centroidal variant of Algorithm 2.

bisectors and intersect them with the bounding polygon.

The computation of capacity-constrained Voronoi diagrams in Algorithm 2 is solely based on the modification of the weights of the sites. Hence, elongated regions appear in the case of the AW distance function, and sites that do not reside within their corresponding Voronoi region appear in the case of the PW distance function. Even though such Voronoi diagrams are valid and fulfill the capacity constraint, they may be unfavorable with regard to the intended application. The solution to both problems is to additionally modify the locations of the sites to achieve more homogeneous distributions. Here, the best results are achieved if the sites reside in the centroids of their corresponding regions. In such case, the resulting diagram becomes a *centroidal capacity-constrained Voronoi diagram*.

A straightforward approach to achieve the additional constraint of centroidal sites is to directly integrate Lloyd's method into Algorithm 2. We therefore insert the additional step of relocating the site s_i to the centroid m_i of the corresponding region $V(s_i, w_i)$ between line 4 and 5 of the algorithm. The advantage of this additional relocation step is that the sites will now be homogeneously distributed within the underlying space. The disadvantage is that this additional operation deteriorates the convergence of the algorithm. A small site location change may entail a couple of other relocations that result in a dramatic increase of the value of $\mathscr{D}$. Nevertheless, experiments showed that the algorithm still exhibits a reliable convergence towards the global minimum $\mathscr{D} = 0$.

Four examples for the generation of centroidal capacity-constrained Voronoi diagrams are presented in Figure 7. These examples are based on the same data as the previous four examples, whereas, in contrast, their computation now included the additional site relocation step. All four examples exhibit homogeneous site distributions with very compact Voronoi regions, and sites that always reside in the region centroids. An overview of the computation time, the number of iterations, et cetera, is given in Figure 9. By analyzing the development of the function $\mathscr{D}$ for these four examples in Figure 8, it becomes apparent that during the first part of the computation, the centroidal variant converges faster than the original non-centroidal algorithm. Furthermore, during these iterations, the permanent change of site locations results in an alternation of the value of $\mathscr{D}$. In the second part, the distribution becomes stable, sometimes with a last large change of various site locations that results in a major increase of $\mathscr{D}$. Afterwards, the distribution is stable, only minor site location changes occur, and the weights are adjusted until the capacity constraint is finally fulfilled, showing a slightly slower convergence than the original non-centroidal algorithm.

The computation times of the examples in this section are significantly smaller than those of the ordinary distance function examples in the previous section. Actually, the weighted approach even allows to generate capacity-constrained Voronoi diagrams for large datasets with thousands or even millions of sites. For example, the computation of a dataset with one million sites of equal capacity using

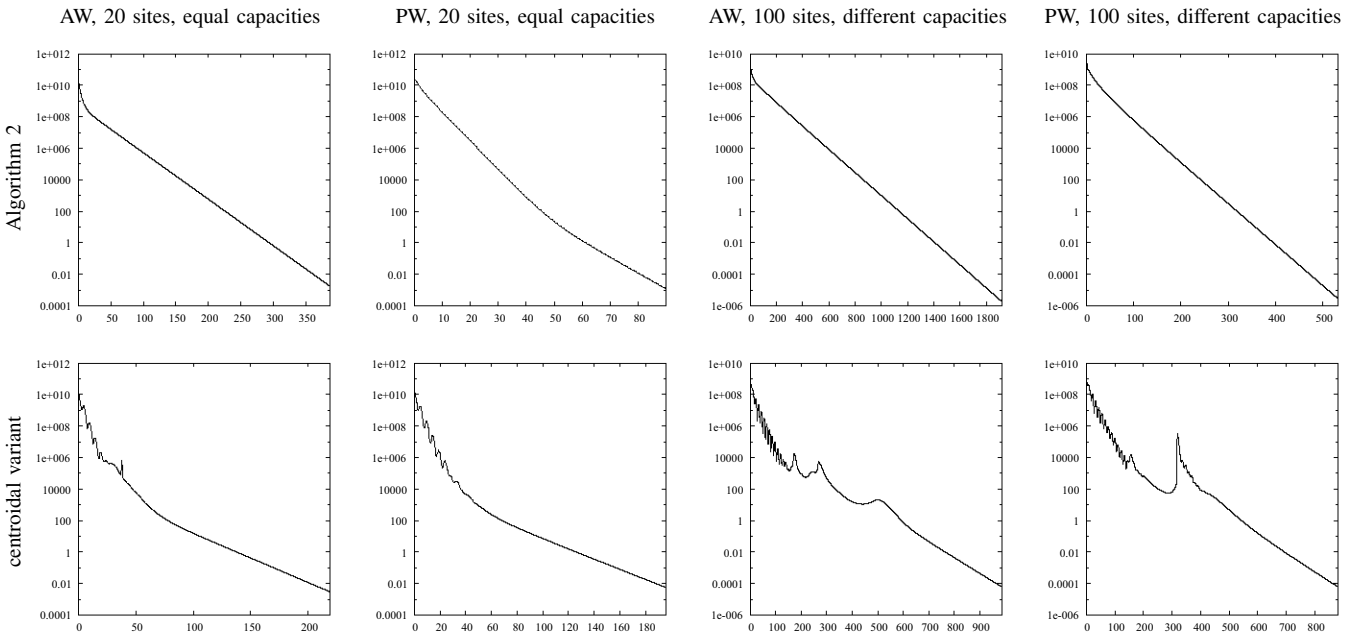


Figure 8. Development of the function $\mathscr{D}$ for each iteration in the weighted examples in Figure 6 and Figure 7.

dataset	computation time	iterations	Voronoi updates	minimum weight	maximum weight
AW, 20 sites, equal capacities	4 sec	387	28,379	−322	300
PW, 20 sites, equal capacities	0.1 sec	90	8,364	−138,646	136,133
AW, 100 sites, different capacities	152 sec	1921	527,475	−414	291
PW, 100 sites, different capacities	5 sec	532	141,572	−101,732	−61,562
centroidal, AW, 20 sites, equal capacities	3 sec	219	17,474	−53	22
centroidal, PW, 20 sites, equal capacities	0.2 sec	196	13,284	−22,045	−14,685
centroidal, AW, 100 sites, different capacities	96 sec	988	352,274	−50	2
centroidal, PW, 100 sites, different capacities	7 sec	882	238,472	−6,951	19

Figure 9. Overview of the computation time, the number of iterations, the number of necessary Voronoi diagram updates during the root finding, and the minimum and maximum weights of the results for all weighted examples.

the PW distance function was performed in less than one day. The reason for this significantly reduced computational effort is the simplification of the optimization from a function for all sites to a much simpler function for just one single site. The resulting optimization for one single site is reduced to a simple root finding that only incorporates the update and extraction of one single Voronoi region. The computational time complexity for this operation is $O(\log n)$, reasoned by the update of the Voronoi diagram. The time complexity for a complete iteration is $O(n \log n)$. In contrast, the ordinary case included the extraction of all Voronoi regions in the diagram with a computational time complexity of $O(\log n + n)$ for a single function evaluation.

IV. Conclusion

We presented two approaches for the generation of capacity-constrained Voronoi diagrams in continuous spaces based on ordinary and/or weighted distance functions. Even though we cannot present a proof for the convergence of these algorithms, our experiments showed their reliable convergence towards arbitrarily precise capacity-constrained Voronoi diagrams. The computational complexity class of our approaches is $O(n^2)$ for the ordinary case, and $O(n \log n)$ for the weighted case. Unfortunately, the iterative structure of our algorithms and the costly function evaluation actually results in a high computational effort for large sets of sites. However, especially our weighted approach enables the generation of capacity-constrained Voronoi diagrams with thousands or even millions of sites. In future work, we will try to further improve the runtime of our algorithms, and evaluate the application of other ordinary and/or weighted distance functions.

References

[1] M. Balzer, T. Schlömer, and O. Deussen, "Capacity-constrained point distributions: A variant of Lloyd's method," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009)*, vol. 28, no. 3, August 2009, to appear.

[2] M. Balzer and O. Deussen, "Voronoi treemaps," in *Proceed-*

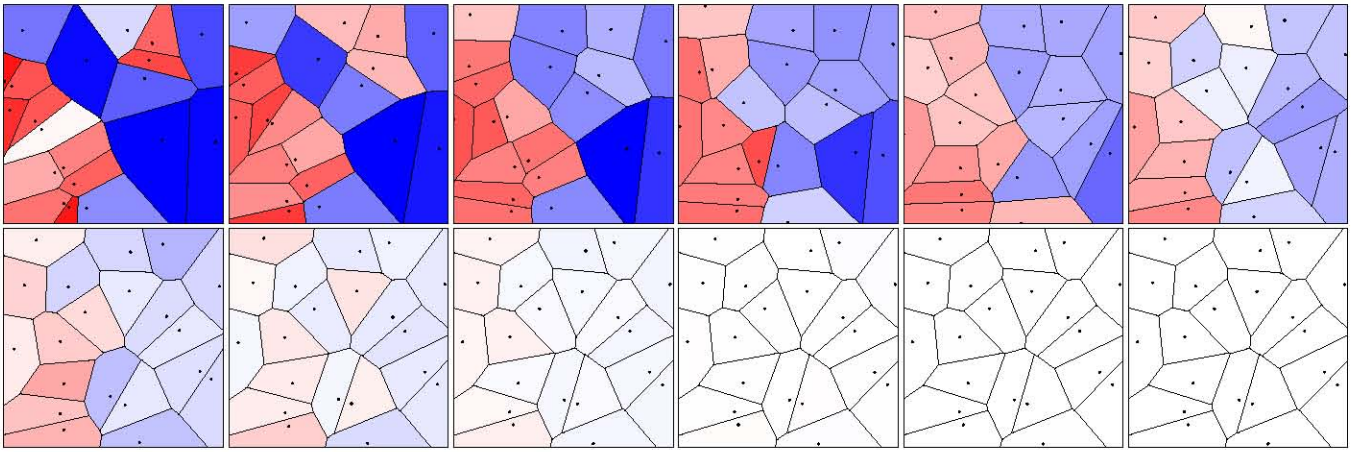


Figure 10. Development of the site locations during the computation of the capacity-constrained ordinary Voronoi diagram in Figure 2(a): initial state, iterations 1–6, 8, 10, 15, 20, result at iteration 35. The colors denote the relative capacity error of each Voronoi region, where blue regions are too large and red regions are too small.

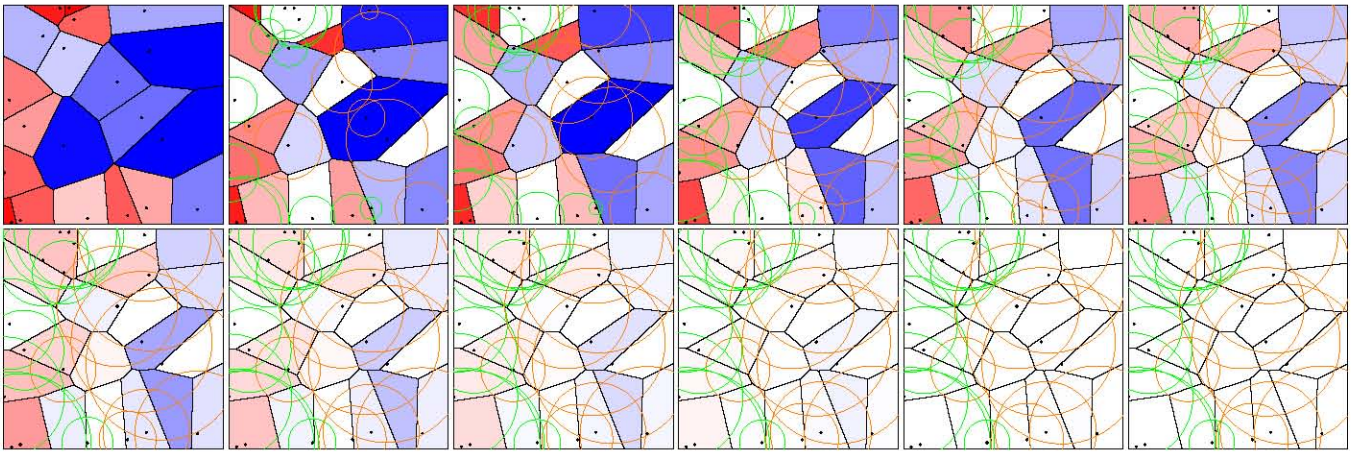


Figure 11. Development of the weights during the computation of the capacity-constrained weighted Voronoi diagram in the lower left of Figure 6: initial state, iterations 1, 2, 4, 6, 8, 10, 15, 20, 30, 50, result at iteration 90. The circles illustrate the weights, where green circles denote positive and red circles denote negative weights.

ings of the IEEE Symposium on Information Visualization. IEEE Computer Society, October 2005, pp. 49–56.

[3] F. Aurenhammer, F. Hoffmann, and B. Aronov, "Minkowski-type theorems and least-squares clustering," *Algorithmica*, vol. 20, no. 1, pp. 61–76, 1998.

[4] T. Ohyama, "Division of a region into equal areas using additively weighted power diagrams," in *Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering*, July 2007, pp. 152–158.

[5] M. Balzer and D. Heck, "Capacity-constrained Voronoi diagrams in finite spaces," in *Proceedings of the 5th Annual International Symposium on Voronoi Diagrams in Science and Engineering*, September 2008, pp. 44–56.

[6] R. Reitsma, S. Trubin, and E. N. Mortensen, "Weight-proportional space partitioning using adaptive Voronoi diagrams," *Geoinformatica*, vol. 11, no. 3, pp. 383–405, 2007.

[7] F. Aurenhammer and R. Klein, "Voronoi diagrams," in *Handbook of Computational Geometry*. Elsevier Science B.V., December 1999, ch. 5, pp. 201–290.

[8] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. John Wiley and Sons Ltd., May 2000.

[9] S. P. Lloyd, "Least square quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.

[10] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, December 1999.

[11] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.