Stable-Matching Voronoi Diagrams

David Eppstein
University of California, Irvine

21st Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCG^3)

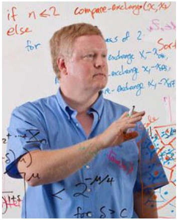
Ateneo de Manila University, Philippines, 2018

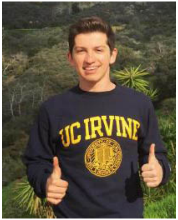
Acknowledgement

This is joint work with:

Gill Barequet

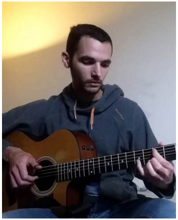
Mike Goodrich





Doruk Korkmaz

Nil Mamano



and is based on papers presented at IWCIA 2017, SIGSPATIAL 2017, LATIN 2018, and ICALP 2018

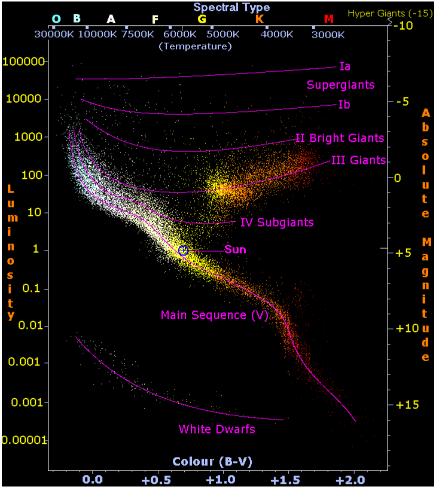
I. Background

Geometric clustering

Goal: Given points in the plane, group them into "meaningful" clusters

Sometimes, # clusters is given, other times it must be inferred

Classical example: Hertzsprung–Russell diagram of stars, plotted by color and brightness

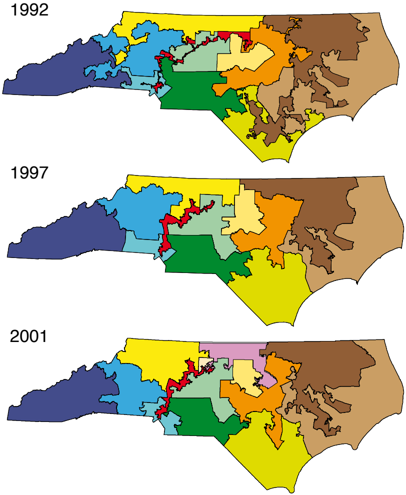


Beyond data analysis

Grouping geometric points into well shaped subsets also has real-world applications

E.g. in political redistricting,

clusters of places ⇔ government officeholders



Optimal clustering

Define a quality measure on clustering:

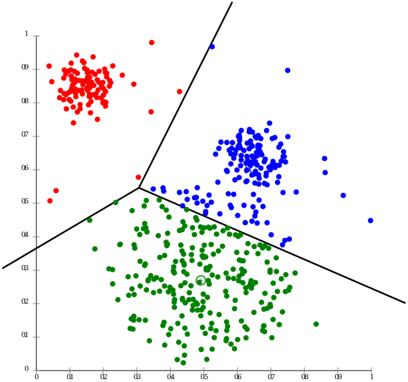
- max diameter of a cluster
- max circumradius of a cluster
- average distance between points in same cluster
- max distance between points in same cluster
- min distance between points in different clusters
- min perimeter of boundaries between clusters
- etc etc

Search for the clustering that optimizes that measure

Voronoi clustering

Choose center points for each cluster

Assign each one the region closer to it than other centers



Can be optimal for some measures

(with the right placement of center points)

Facility location

Distribute facilities (points) to best serve surrounding regions

Map of US Starbucks locations from https://www.redliondata.com/chain-store-maps-tim-hortons-vs-starbucks/

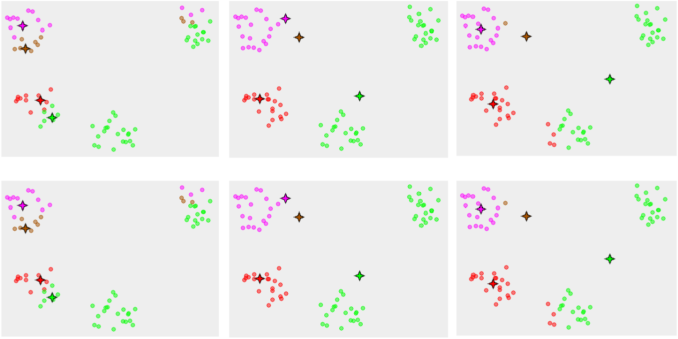
Typically, each facility serves its nearest neighbors So the regions it serves are Voronoi clusters

EM / Lloyd / k-means

Solve both clustering and facility location (and even finite element mesh smoothing!) by shifting cluster centers in Voronoi clustering

Repeat:

- Compute the Voronoi clustering for the current centers
- Move each center to the centroid (or circumcenter) of its new cluster

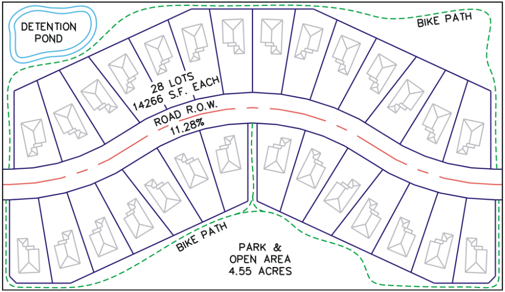


II. Definition and basic properties

Capacity / size constraints

Political redistricting requires each region to have equal population

Load-balanced data distribution, property subdivision require each region to have equal area



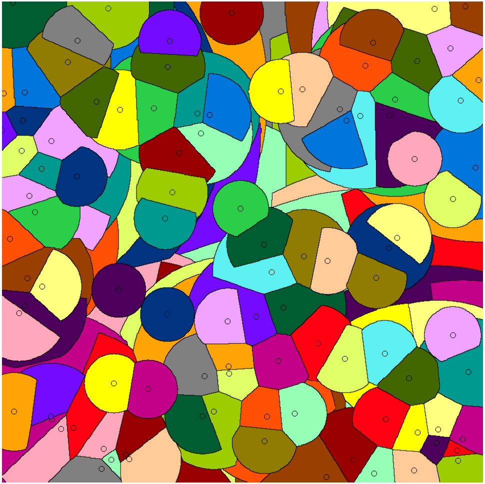
Free Art License image SubdivisionCoving.svg by Zephram Stark from Wikimedia commons

Capacity constraints are also standard in facility location problems

How can we achieve this?

Stable-marriage Voronoi diagrams

Given centers, match regions of given areas to each center so that no unmatched point & center are closer than their matches



Why “stable marriage”?

Classical stable marriage: n men, n women, each with preferences

Goal: Pair men and women so no unmatched pair likes each other better than their matches (avoid unstable pairs)

Mass wedding at Unification Church, 2013, from https://www.cnn.com/2013/02/17/asia/gallery/mass-wedding/index.html

Widely used e.g. to assign medical students to residencies

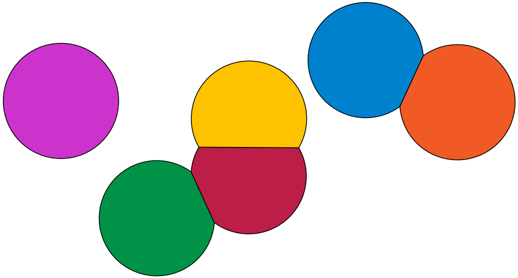
Here, we are matching points to centers in the same way with Euclidean distances as preferences

Existence and uniqueness

[Hoffman, Holroyd, and Peres, 2006]

Grow circles around each center at equal rates

Match regions to the first circle that covers them



Stop growing when the target region area has been assigned

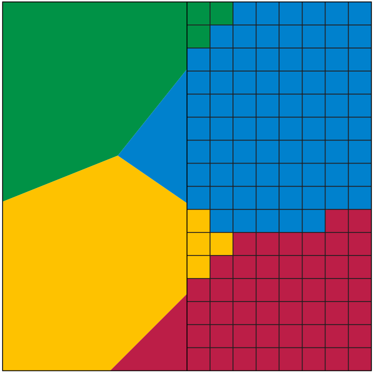
All region–center matches are stable and forced

III. Pixelation

But how can we calculate it?

Our first approach [IWCIA 2017]: Pixelate!

Partition the area we are trying to partition into a grid of pixels



Find a stable marriage between pixels and cluster centers

Strawman: Gale–Shapley algorithm

Each center ⇒ a number of men equal to its capacity

Each pixel ⇒ one woman

Repeat:

- Each single man proposes to the nearest woman he hasn't already proposed to
- Each woman agrees to marry the nearest man who proposes to her (possibly dumping an agreement she made earlier)

Time to match an $n \times n$ grid: $O(n^4)$

Time to find priorities: bigger?

DE, Tanaka Farms, 2003

How to prioritize the pixels

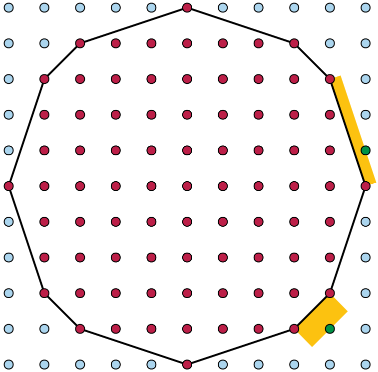
Maintain convex hull of pixels generated so far

Next pixel is offset from a convex hull edge at distance 1/length(edge)

Maintain $O(1)$ candidates/edge prioritized using a bucket queue

$O(1)$ time per pixel

Can stream sorted pixels of $n \times n$ grid in space $O(n^{2/3})$



Pixelated circle-growing

For each vector $v \in \mathbb{Z}^2$ (in sorted order by length):

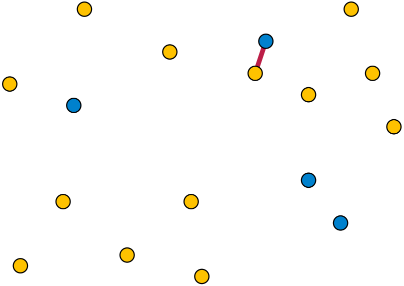
> For each center c that has not reached capacity:
>
> > If $c + v$ is inside the grid and not already assigned:
> >
> > > Match $c + v$ to c

With C centers in an $n \times n$ grid, worst case time is $O(Cn^2)$

A. V. Tyranov, *Boy with Bubbles*, 1856

Bichromatic closest pairs

Repeatedly match closest (unmatched pixel, hungry center) pair



Closest pair from two dynamic sets $\Rightarrow$
dynamic nearest neighbors $\times\ O(\log^2 n)$ [Eppstein 1995]

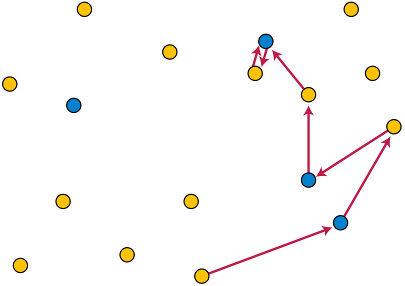
Dynamic nearest neighbor search $O(\log^5 n)$ per operation
[Kaplan et al. 2016, improving Chan 2010]

Total $O(n^2 \log^7 n)$, but complicated and impractical

Neighbor chains

Shave logs by finding mutual nearest neighbors instead of closest pairs (an idea previously used in hierarchical clustering)

Starting at any point, build a stack by repeatedly pushing nearest neighbor of stack top



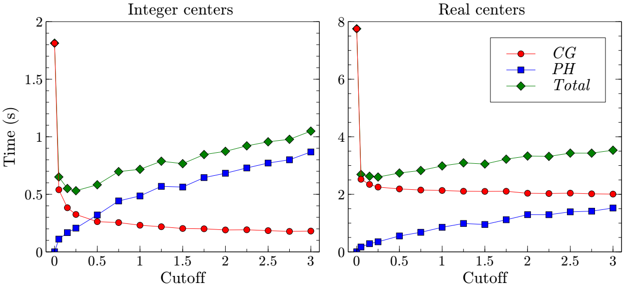
When top two points are mutual nearest neighbors, match and pop

Reduces time to $O(n^2 \log^5 n)$, still impractical

A practical hybrid algorithm

Use circle-growing up to some cutoff radius
(while few of the pixels it finds are unmatched)

Then switch to closest pairs
(slower per pixel but no penalty for unmatched)



IV. Road networks

Stable Voronoi in road networks

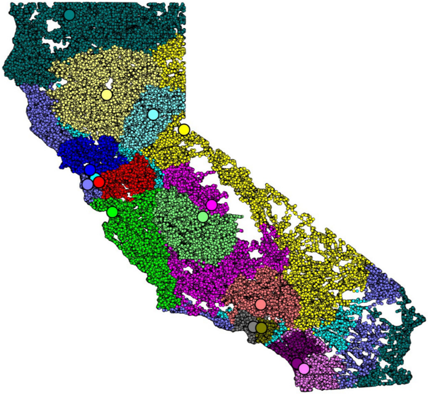
Problem: Cluster real-world geography [SIGSPATIAL 2017]

Difficulties:

- Geographic barriers make distances inaccurate
- We want clusters by population, not area

Solution: Use road network shortest paths!

- Most algorithms still work
- Network complexity stands in for population



Circle-growing in road networks

Run C parallel copes of Dijkstra's shortest path algorithm (one per cluster center)

Match each vertex to the first copy that reaches it

When one center gets enough matches, stop its copy

Total time (for n vertices and C centers) $O(Cn \log n)$

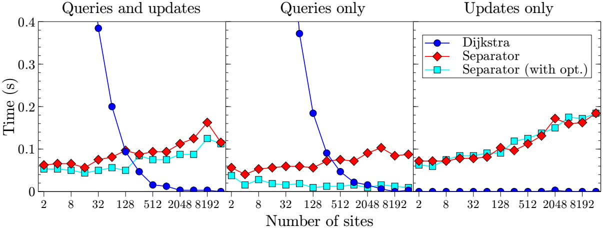
Dynamic nearest neighbors in road networks

Needed for nearest-neighbor chain, potentially useful for other applications e.g. vehicle dispatching [LATIN 2018]

Build separator hierarchy
– Each separator vertex stores priority queue of dynamic points
– Query compares candidate neighbors from separator vertices

Heuristic optimization: sort separator vertices by distance, stop query when distance > best found so far

$O(n^{1/2})$ / query, $O(n^{1/2} \log\log n)$ / update

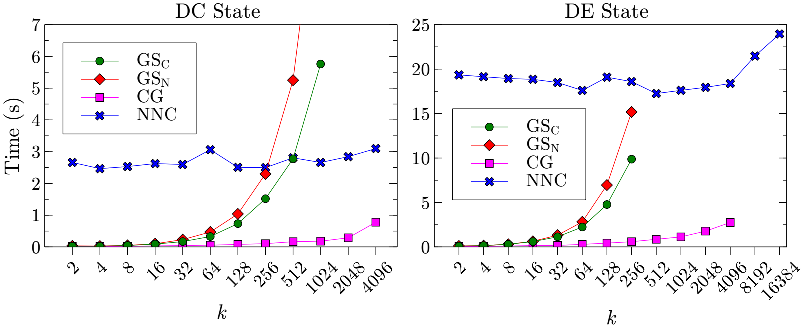


Comparison of algorithms for road networks

Gale–Shapley is only usable for small numbers of clusters

Neighbor chain is $\tilde{O}(n^{3/2})$, independent of # clusters but too slow

Circle growing is best for small to moderate # clusters

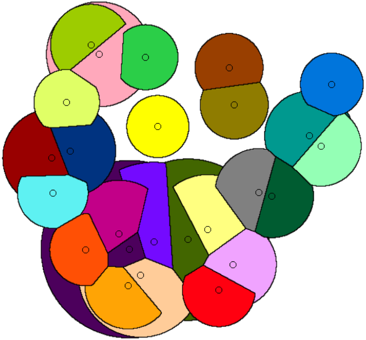


V. Continuous diagrams

Breaking out of the frame

Diagram lives in whole plane, not just a square [ICALP 2017]

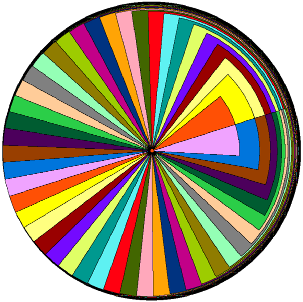
Each center has a capacity (area), usually all equal

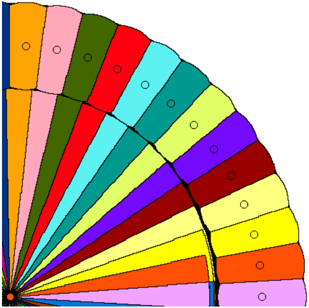


Cell boundaries are lines (between two growing disks)
and circular arcs (when a disk's growth stops)

Lower bound on combinatorial complexity

Place $n/2$ points near center to form rainbow
and $n/2$ points in surrounding circle to take bites out

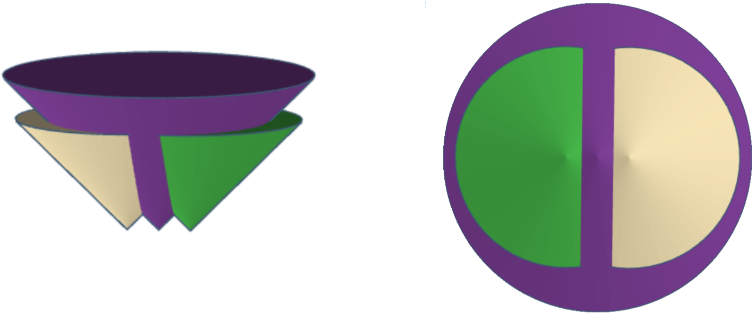




Cells may be disconnected,
with $\Omega(n^2)$ total components
and $\Omega(n^2)$ total complexity

Upper bound from lifting

Grow cones in 3d above plane of centers
Stop growth when each cone has a shadow of the right area



As lower envelope of piecewise algebraic surfaces, diagram has complexity $O(n^{2+\epsilon})$ for any $\epsilon > 0$

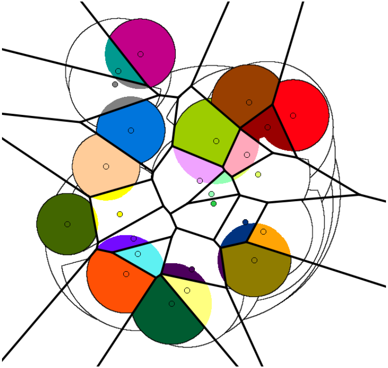
But they are not pseudospheres! If they were, bound would be $O(n^2)$

Paint-by-numbers algorithm

Maintain (classical) Voronoi diagram of still-hungry centers

Repeat:

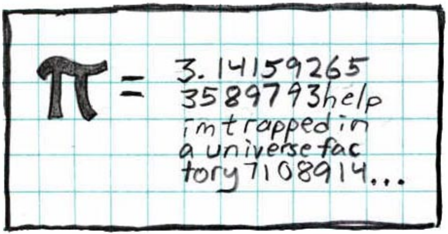
- For each remaining center, find disk s.t. intersection with unmatched points in its cell has target area
- Choose the smallest disk
- Match all points in the disk to their cells
- Remove the disk center from the Voronoi diagram



Algorithmic primitive

Given a convex polygon P (the Voronoi cell of center p),
a target area A, and a set C of disks,
find the radius for which $A = \text{area}\big(B_r(p) \cap (P \setminus \bigcup C)\big)$

This primitive is transcendental
(can't be computed using roots of polynomials)



Paint-by-numbers takes time $O\big(n^3 + n^2 f(n)\big)$
where $f(x)$ is the time for the primitive on inputs of complexity x

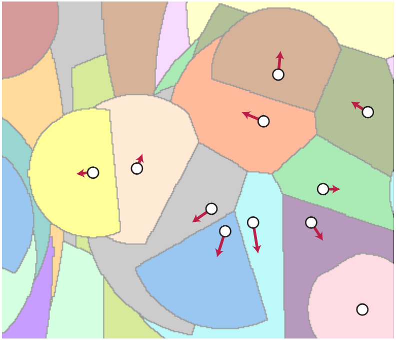
VI. Moving the centers

Optimized center location

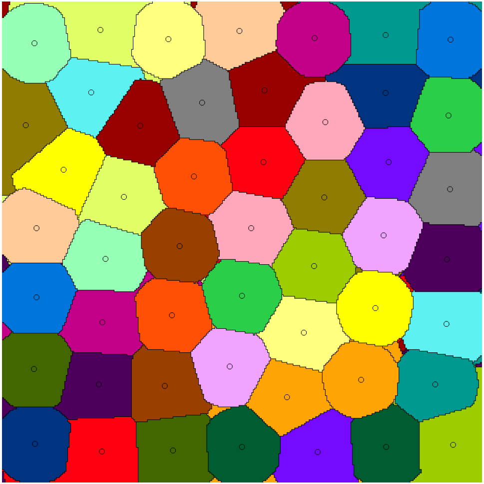
Original goal: Find geographically compact clusters
(e.g. for redistricting)

Modified goal: Place centers to make stable-matching diagram
have connected regions

Approach: Lloyd's algorithm – alternate between constructing the
diagram and moving centers to better locations in their cells



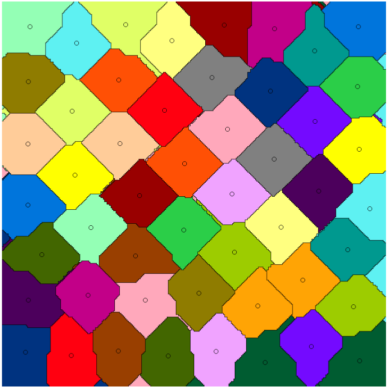
Results of Lloyd's algorithm

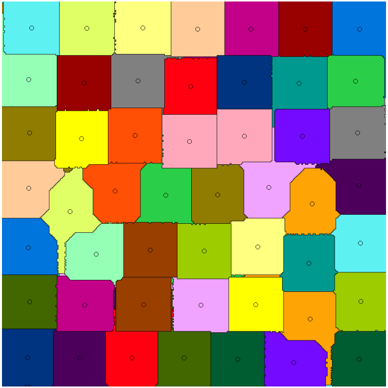


Disconnected cells and odd shapes still exist, but are greatly reduced

Alternative metrics

Also gives interesting results for L_1 and L_∞ metrics





Conclusions and open problems

Conclusions

- Interesting partition of the plane, achieves area constraints but at the expense of connectivity
- Near-linear algorithms for pixelated approximations
- Nontrivial but slower algorithms for planar or near-planar networks
- Cubic-time algorithm (with necessary but nonstandard computational assumptions) for the continuous case
- Near-tight combinatorial complexity bounds

Some open problems

- Tighter upper and lower bounds on combinatorial complexity?
- Subcubic algorithm for the continuous problem?
- Which regions can be partitioned by stable-matching Voronoi diagrams into a given number of *connected* subsets?
  Is this always possible for square regions?
- Approximation? Would letting areas be approximate help make cells connected? What about faster approximate near neighbors?