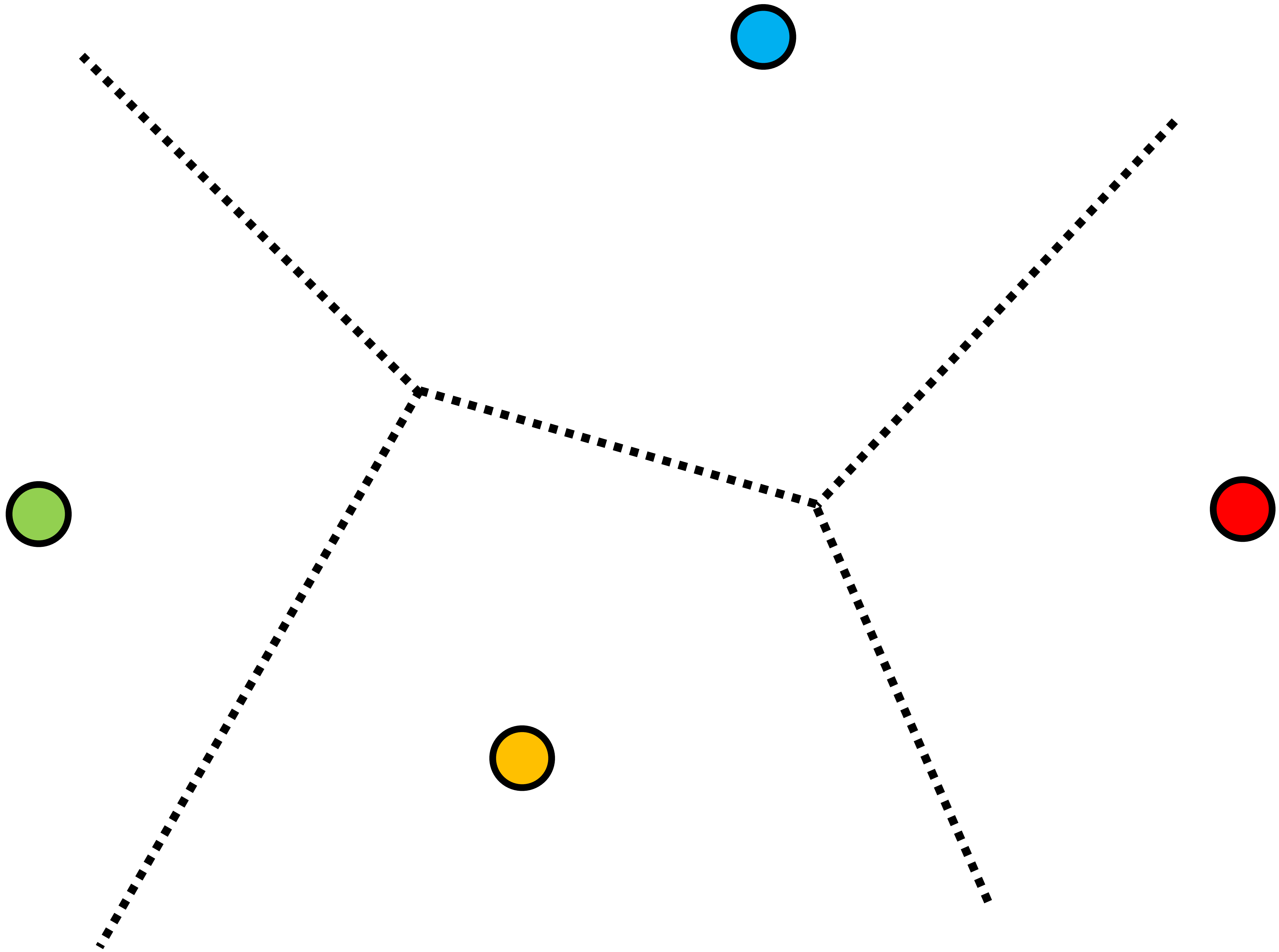
Voronoi Diagrams on Planar Graphs and Computing the Diameter in Deterministic $\tilde{O}(n^{5/3})$ time

Pawel Gawrychowski, Haim Kaplan, Shay Mozes,
Micha Sharir and Oren Weimann

Voronoi diagrams

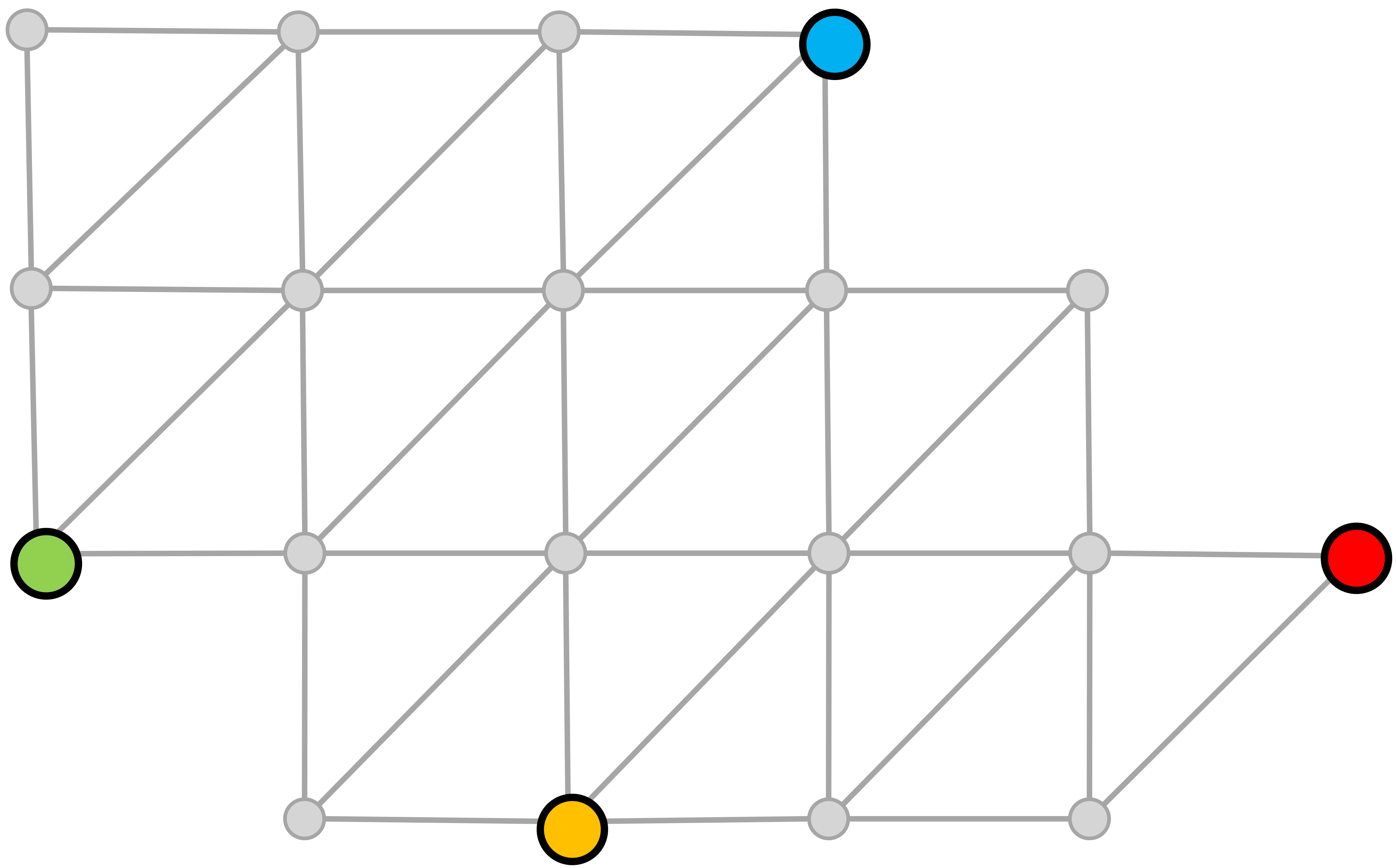


Voronoi
1908

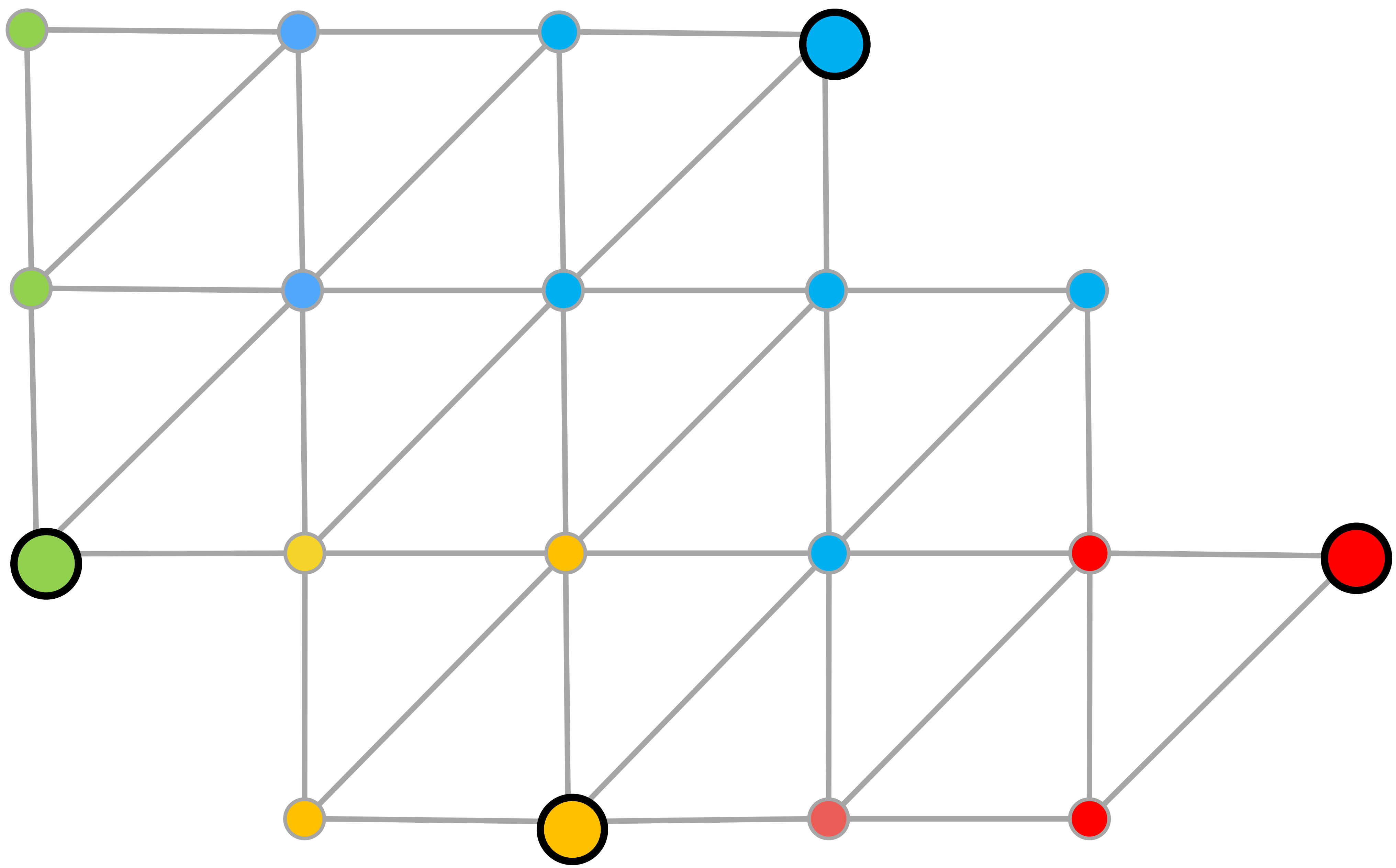
Descartes
1644

Dirichlet
1850

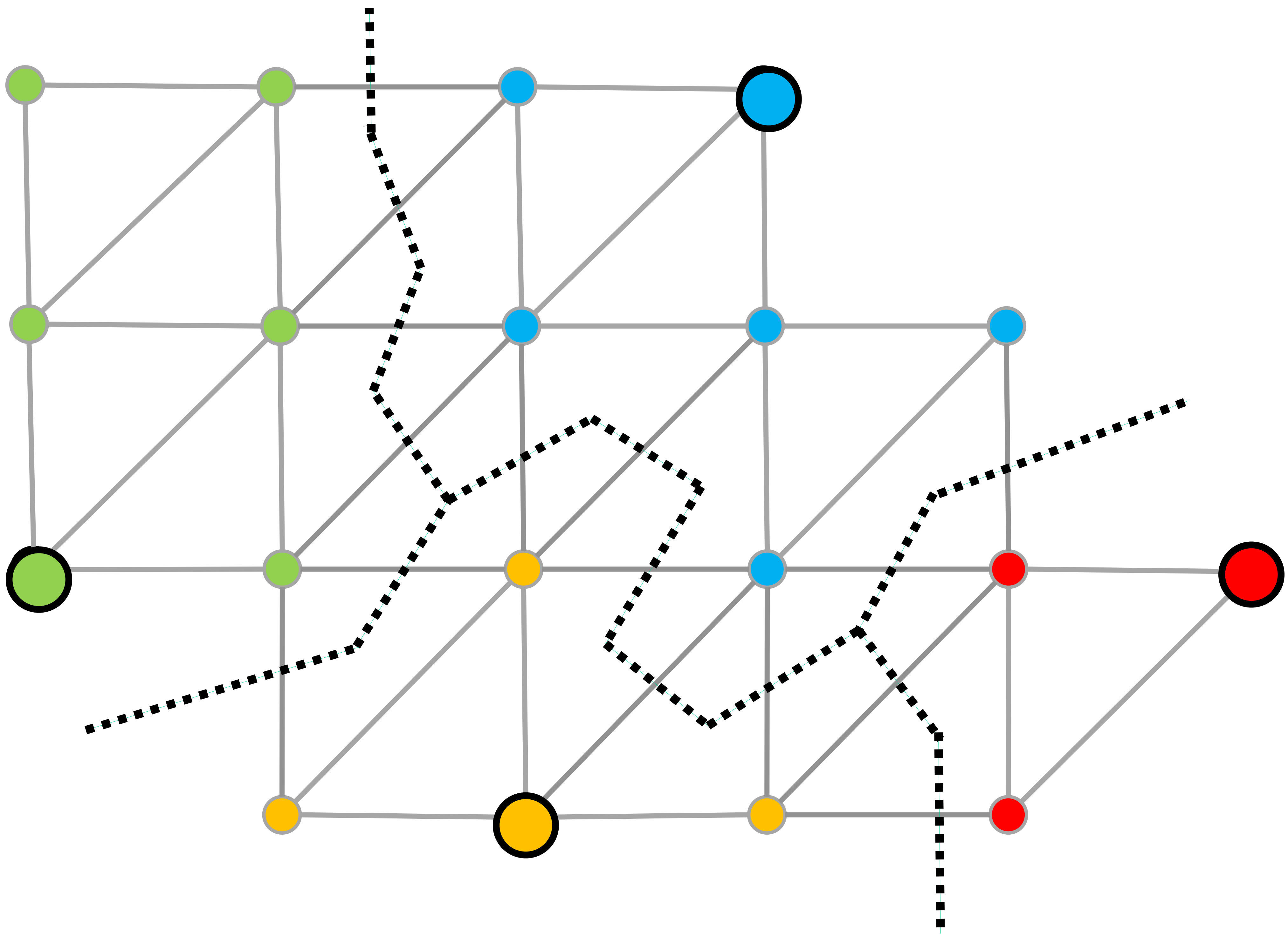
Voronoi diagrams on planar graphs



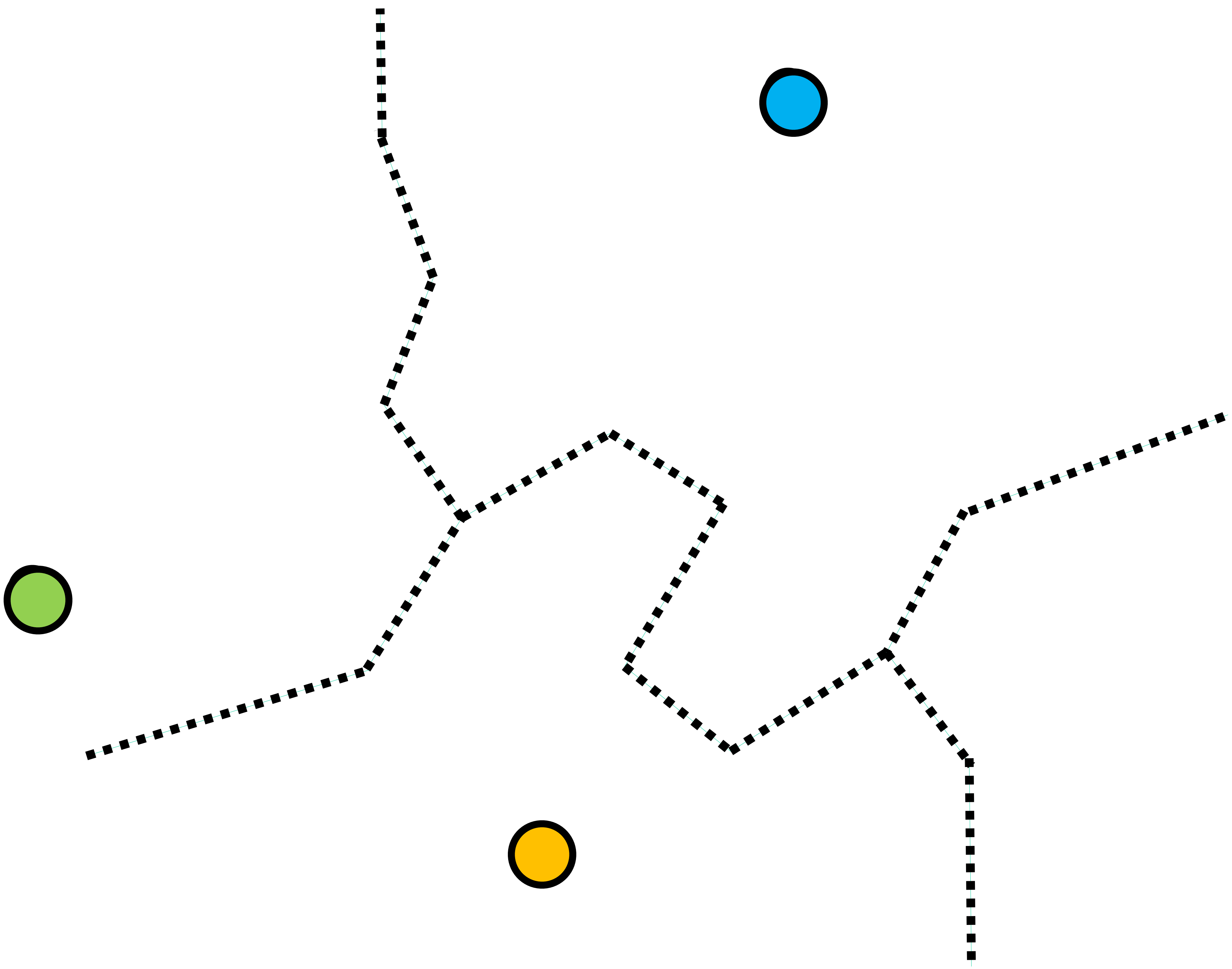
Voronoi diagrams on planar graphs



Voronoi diagrams on planar graphs



Voronoi diagrams on planar graphs



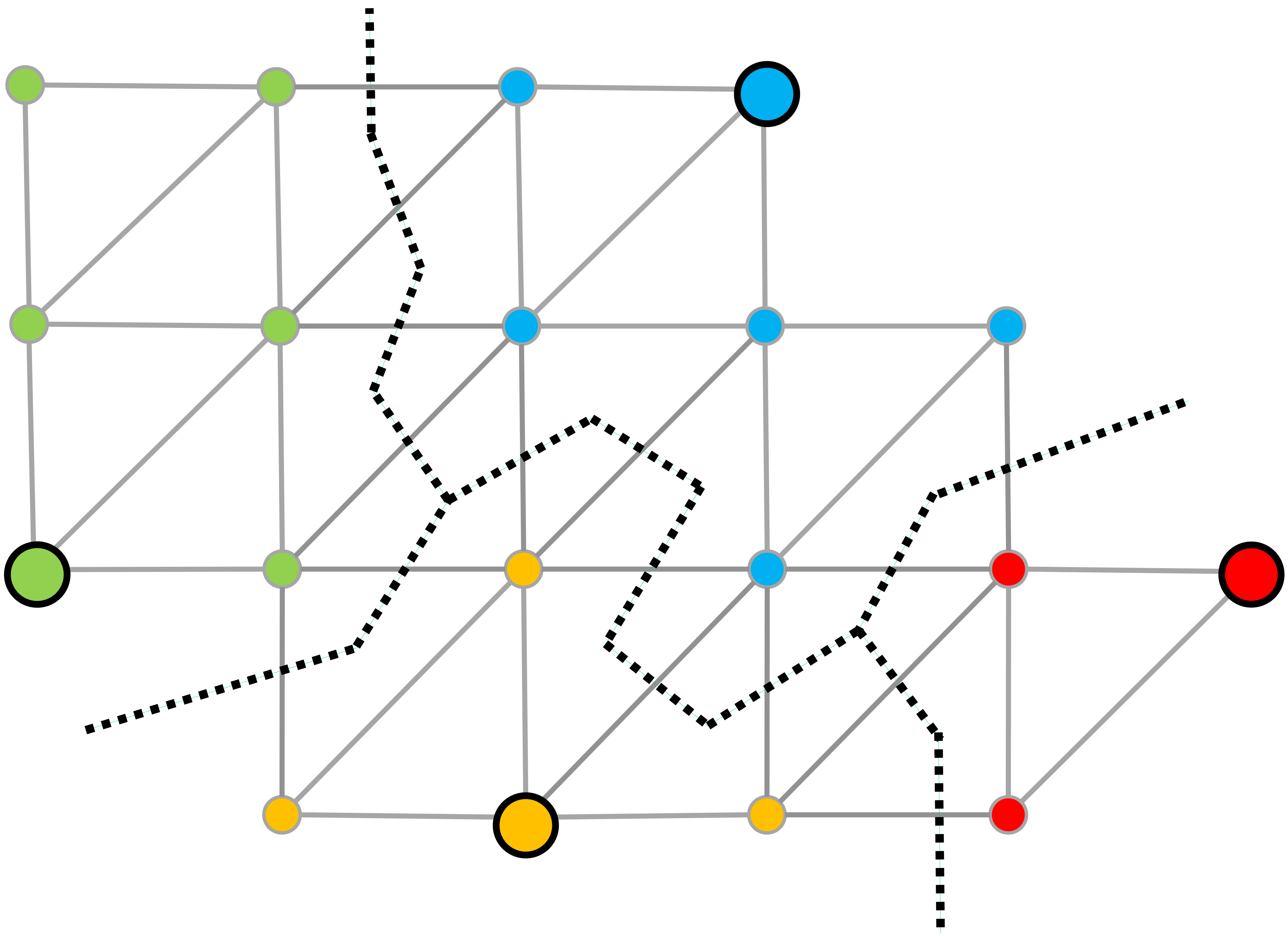
A fresh idea in algorithms for planar graph

- Cabello's breakthrough (best paper in SODA 2017) -
  - can quickly construct Voronoi diagrams on planar graphs
    (using randomized incremental construction of abstract Voronoi diagrams)
  - can use Voronoi diagrams to compute the diameter in sub-quadratic $\tilde{O}(n^{11/6})$ randomized time
- Led to exciting developments in distance oracles for planar graphs [Cohen-Addad et al. FOCS17, next talk]

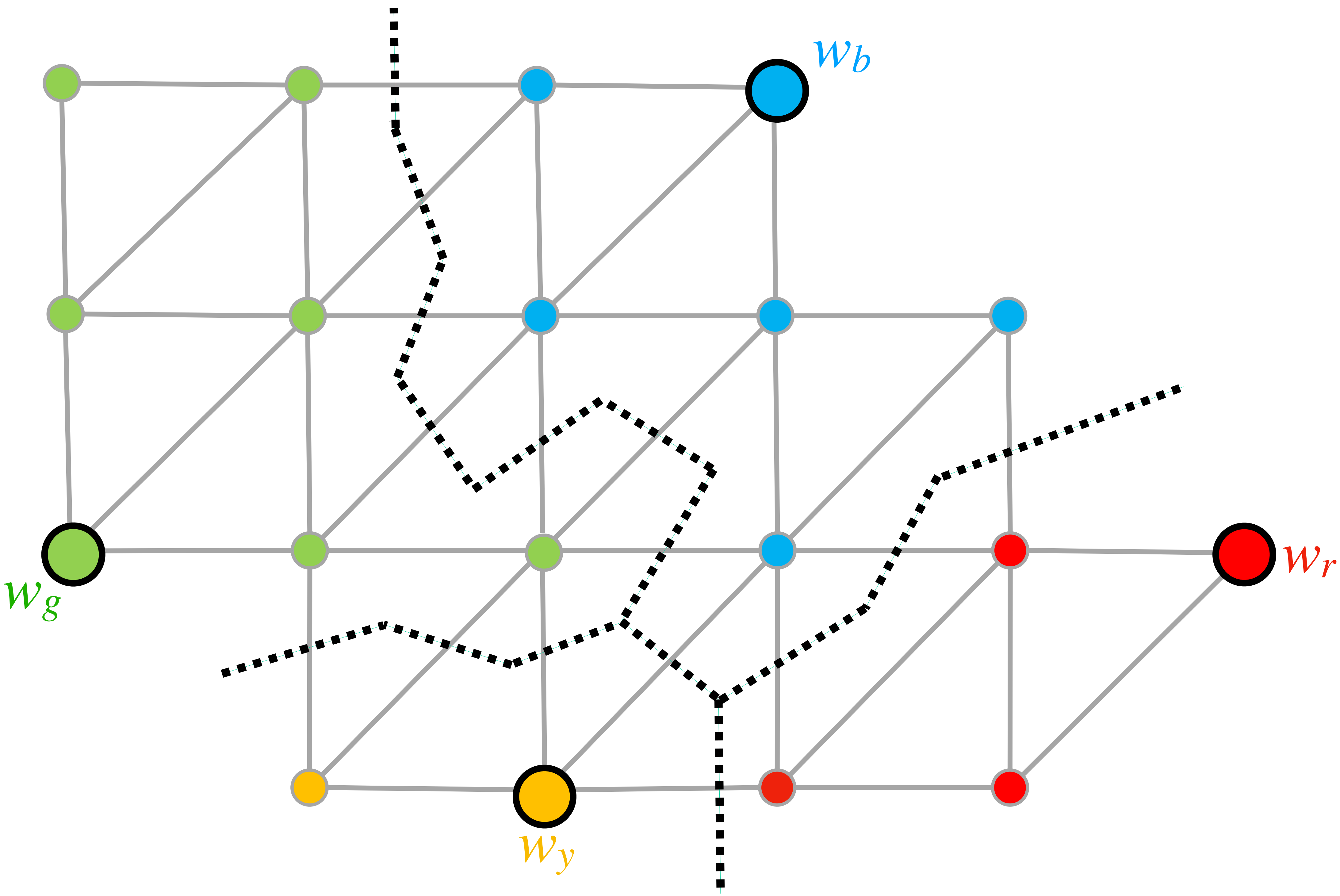
This work

- construction of Voronoi diagrams on planar graphs
  - faster, deterministic, more general
- leads to a faster $O(n^{5/3})$-time algorithm for diameter

Voronoi diagrams on planar graphs

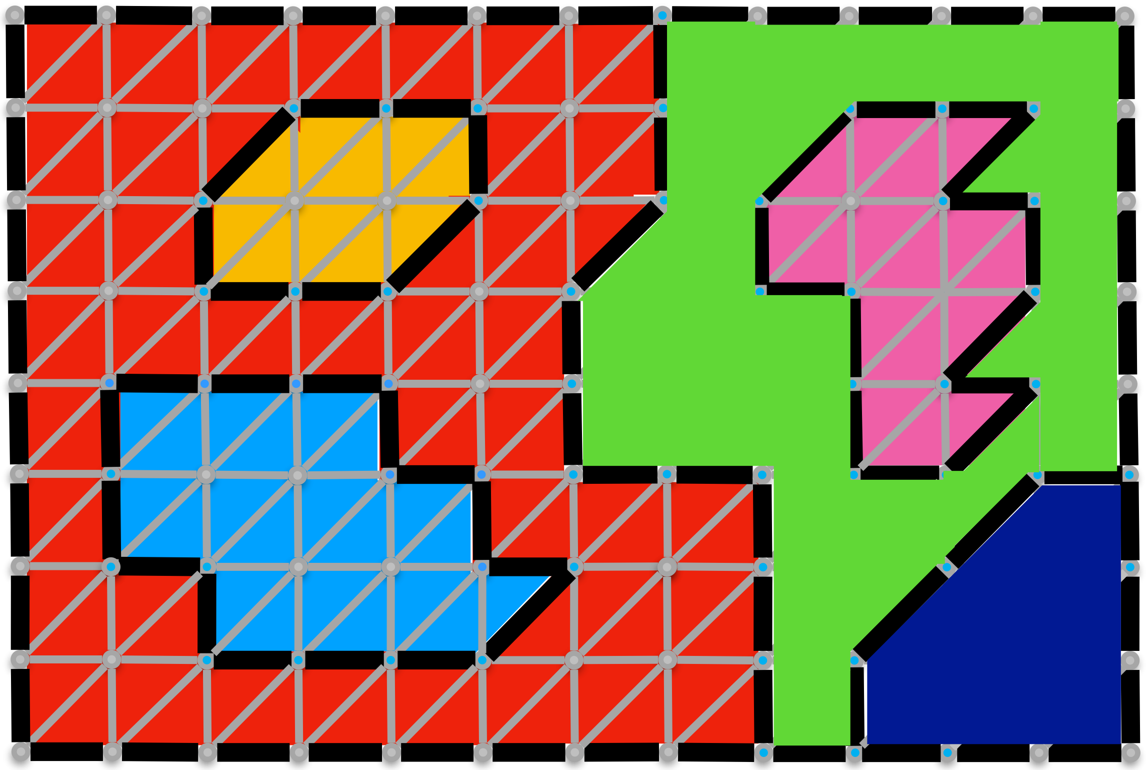


Additively weighted Voronoi diagram



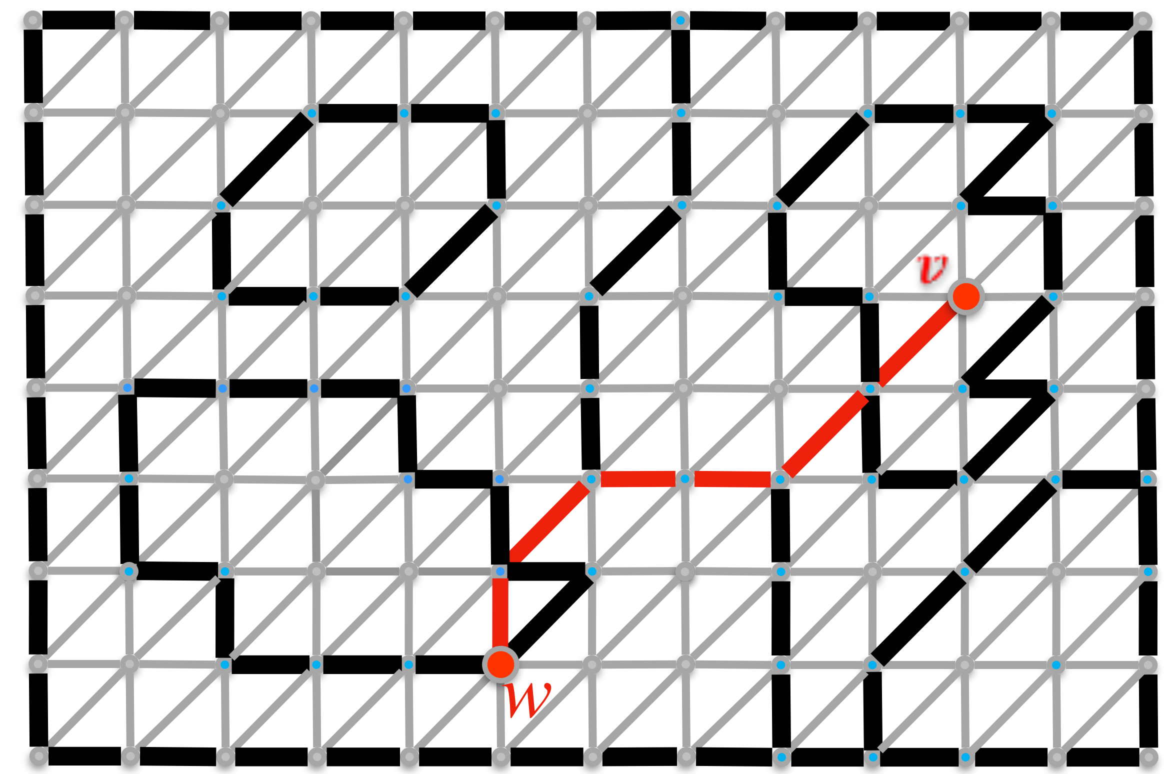
High-level approach for diameter

- compute an r-division: $O(n/r)$ pieces, each with $O(r)$ vertices and $O(r^{1/2})$ boundary vertices



- there are three types of distances:

High-level approach for diameter

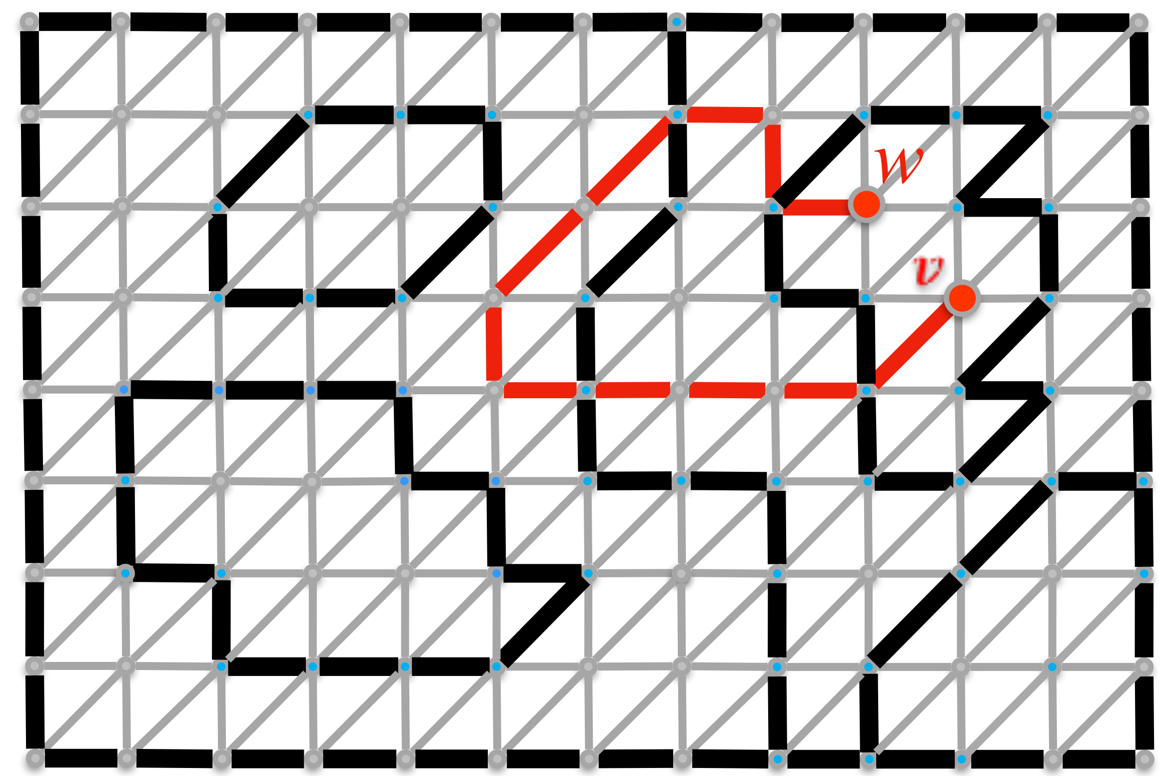


- compute an r-division: $O(n/r)$ pieces, each with $O(r)$ vertices and $O(r^{1/2})$ boundary vertices

- there are three types of distances:
  - between a vertex and a boundary vertex

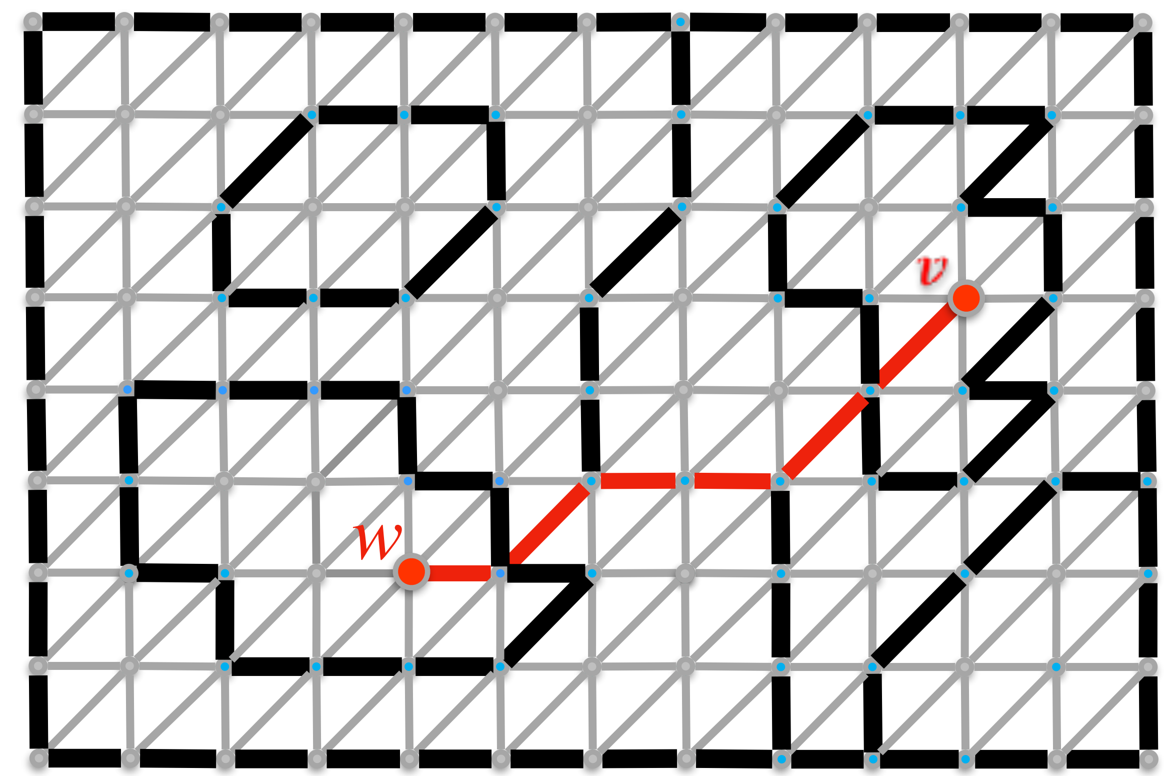
High-level approach for diameter

- compute an r-division: $O(n/r)$ pieces, each with $O(r)$ vertices and $O(r^{1/2})$ boundary vertices



- there are three types of distances:
  - between a vertex and a boundary vertex
  - between two vertices in the same piece

High-level approach for diameter



- compute an r-division: $O(n/r)$ pieces, each with $O(r)$ vertices and $O(r^{1/2})$ boundary vertices

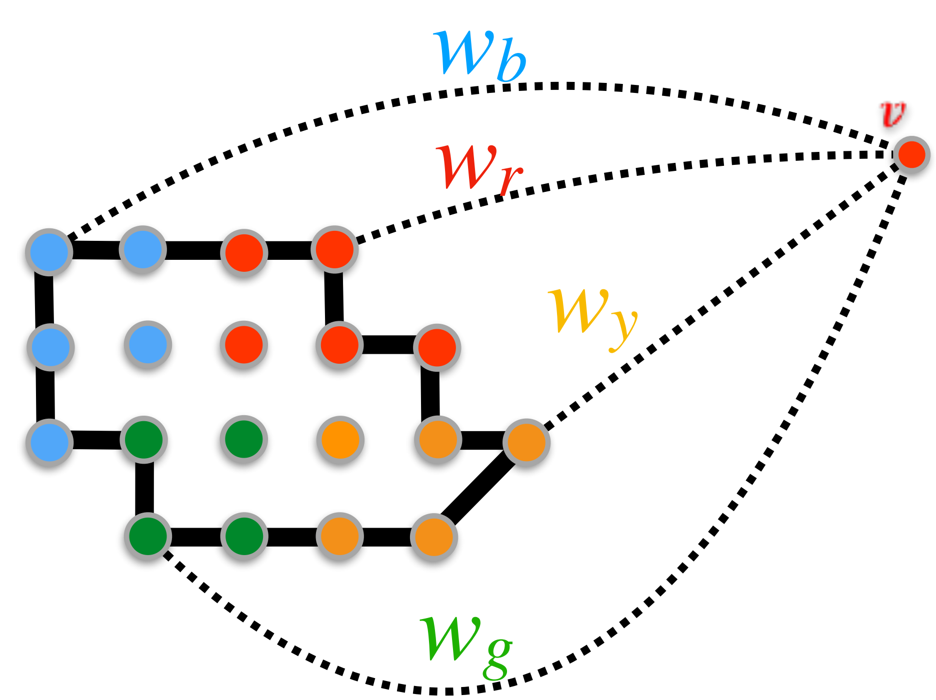
- there are three types of distances:
  - between a vertex and a boundary vertex
  - between two vertices in the same piece
  - between two vertices in different pieces

Dist. between vertices in different pieces

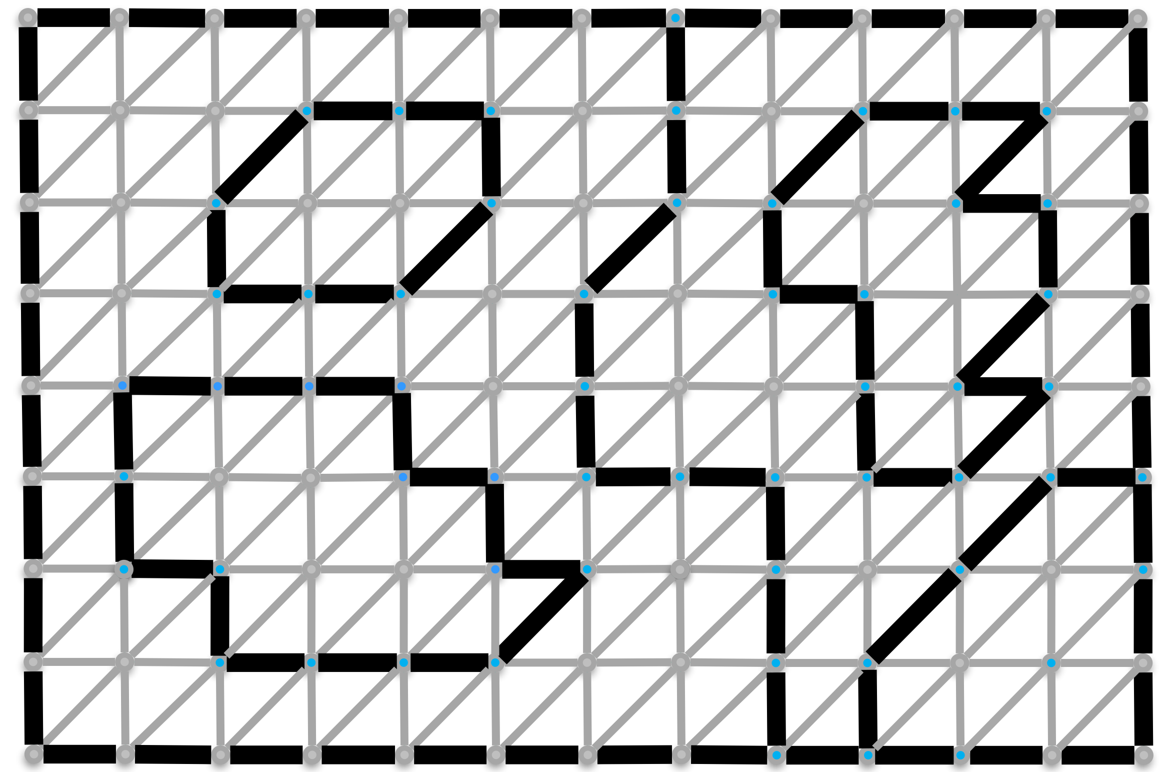
- already computed distances from v to boundary nodes of the other piece
- compute additively weighted Voronoi diagram for the other piece in $\tilde{O}(r^{1/2})$ time
- use Voronoi diagram to return the node furthest from each boundary site in $\tilde{O}(1)$ amortized time per site
- total $\tilde{O}(n \cdot n/r \cdot r^{1/2}) = \tilde{O}(n^2/r^{1/2})$

  # vertices   # pieces

- requires $\tilde{O}(n/r \cdot r^2) = O(nr)$ preprocessing



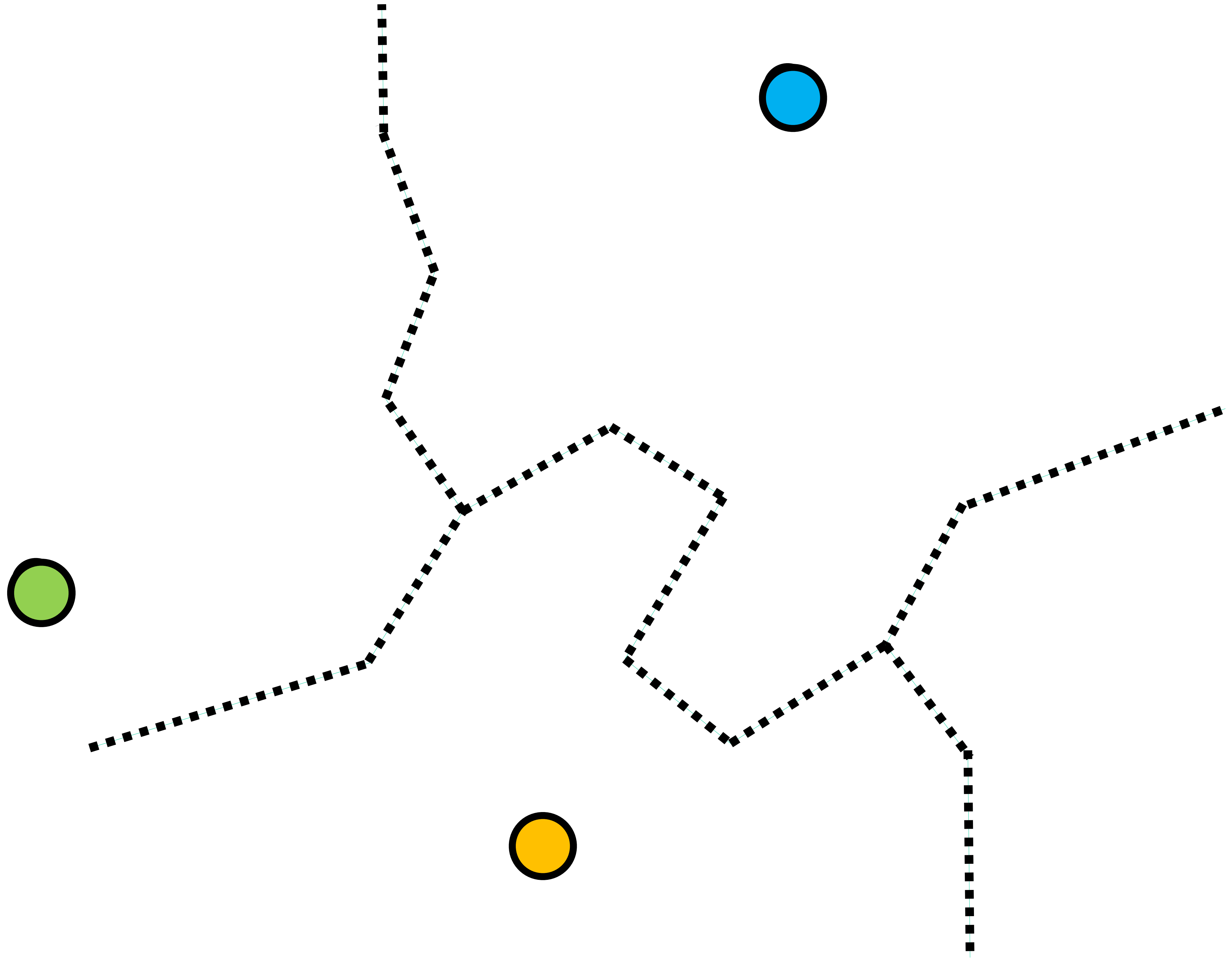
High-level approach for diameter



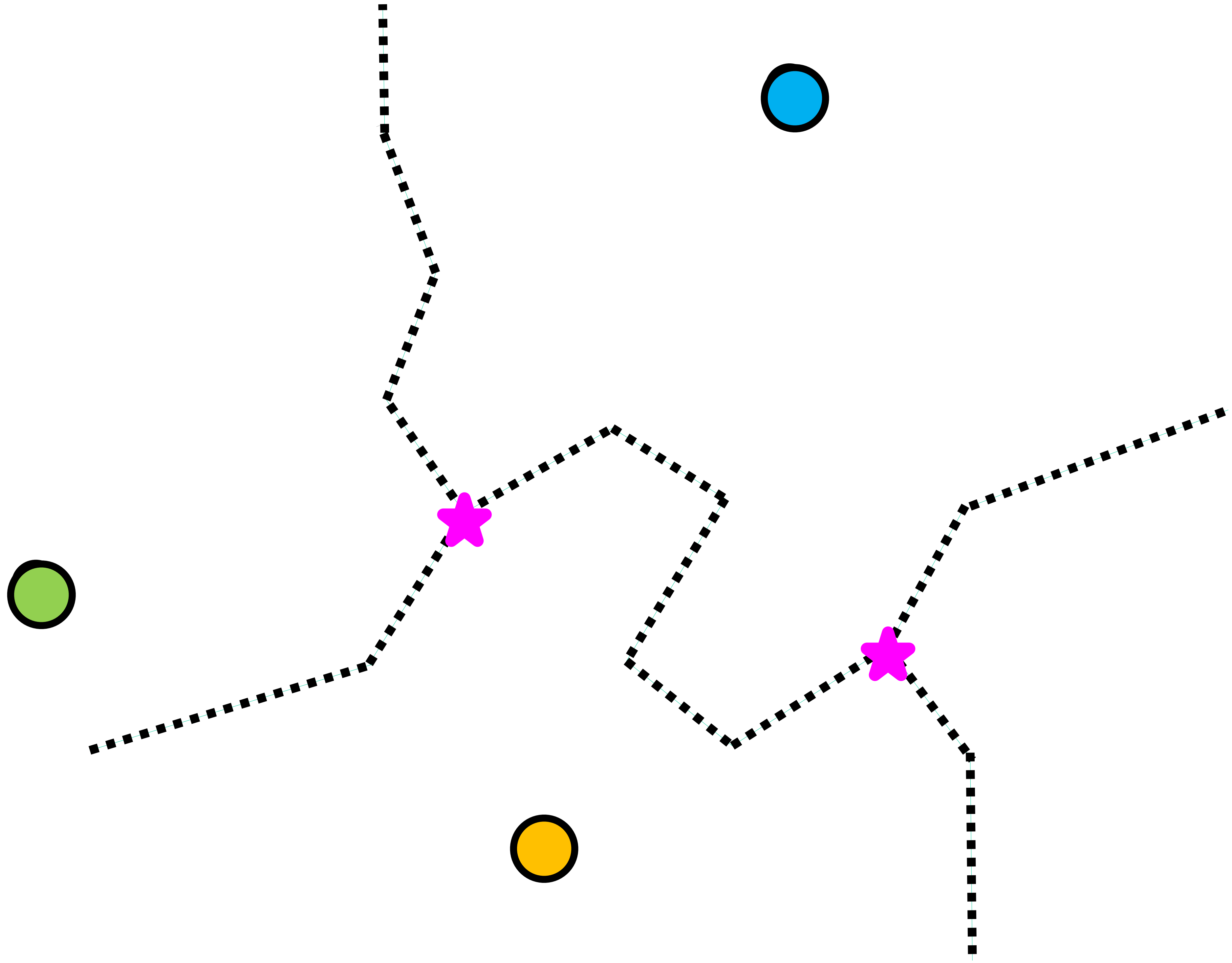
- compute an r-division
- three types of distance:
  - between a vertex and a boundary vertex — $O(n^2/r^{1/2})$ time
  - between two vertices inside the same piece — $O(nr)$ time
  - between two vertices in different pieces — $\tilde{O}(nr + n^2/r^{1/2})$ time

- setting $r = n^{2/3}$ yields total running time of $\tilde{O}(n^{5/3})$

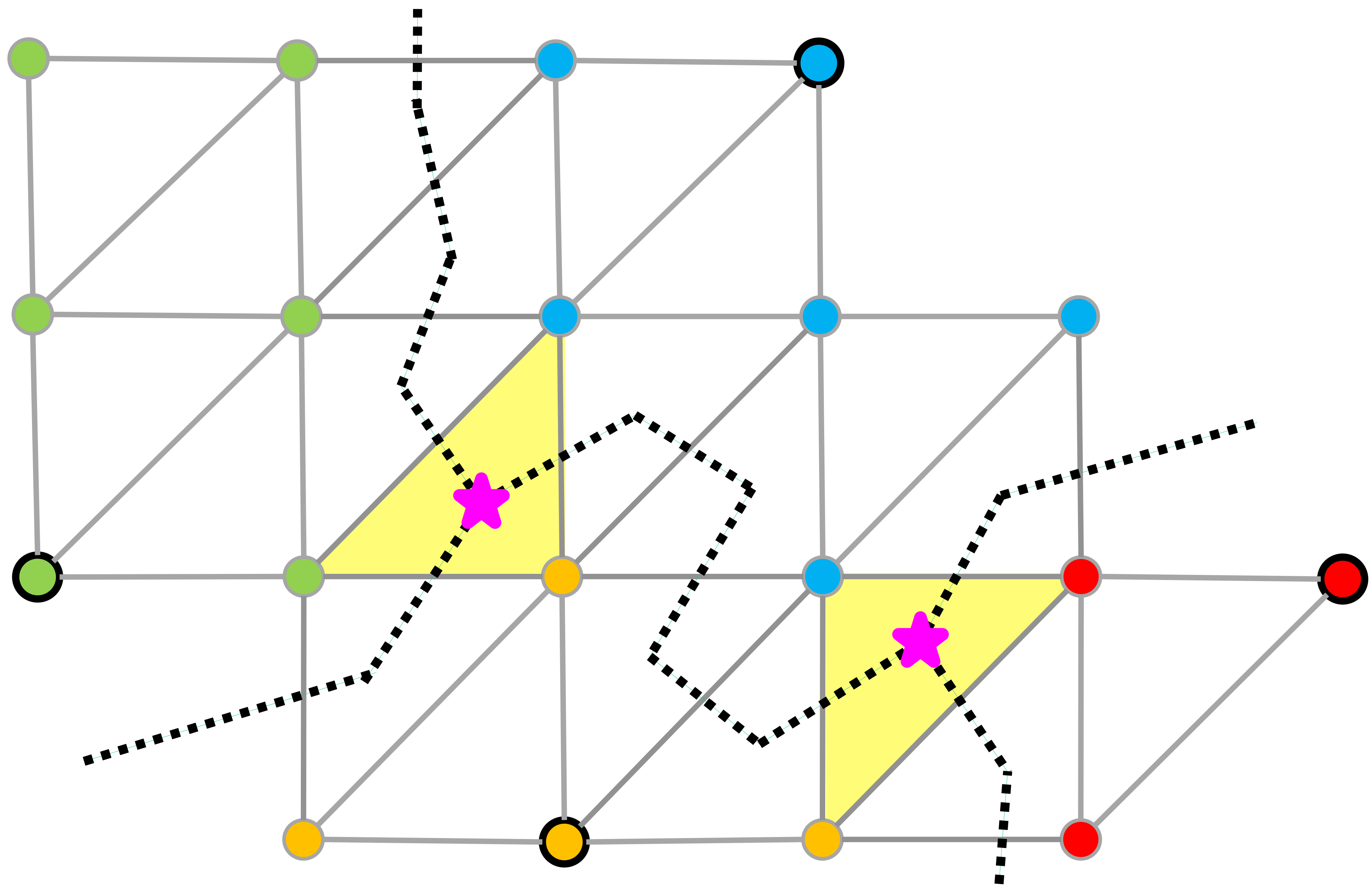
Remainder of the talk: constructing Voronoi diagrams



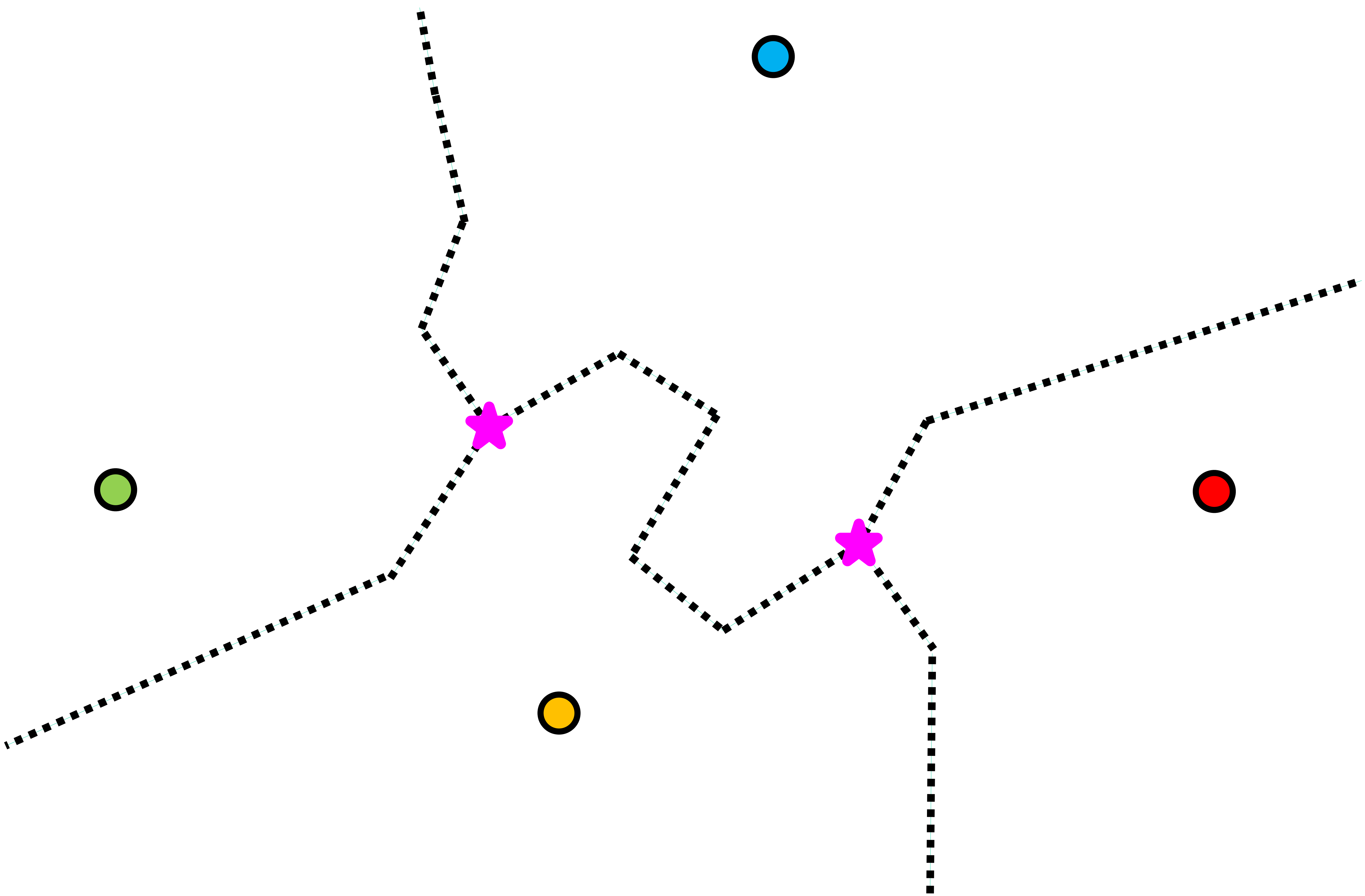
Voronoi vertices - adjacent to three different Voronoi cells



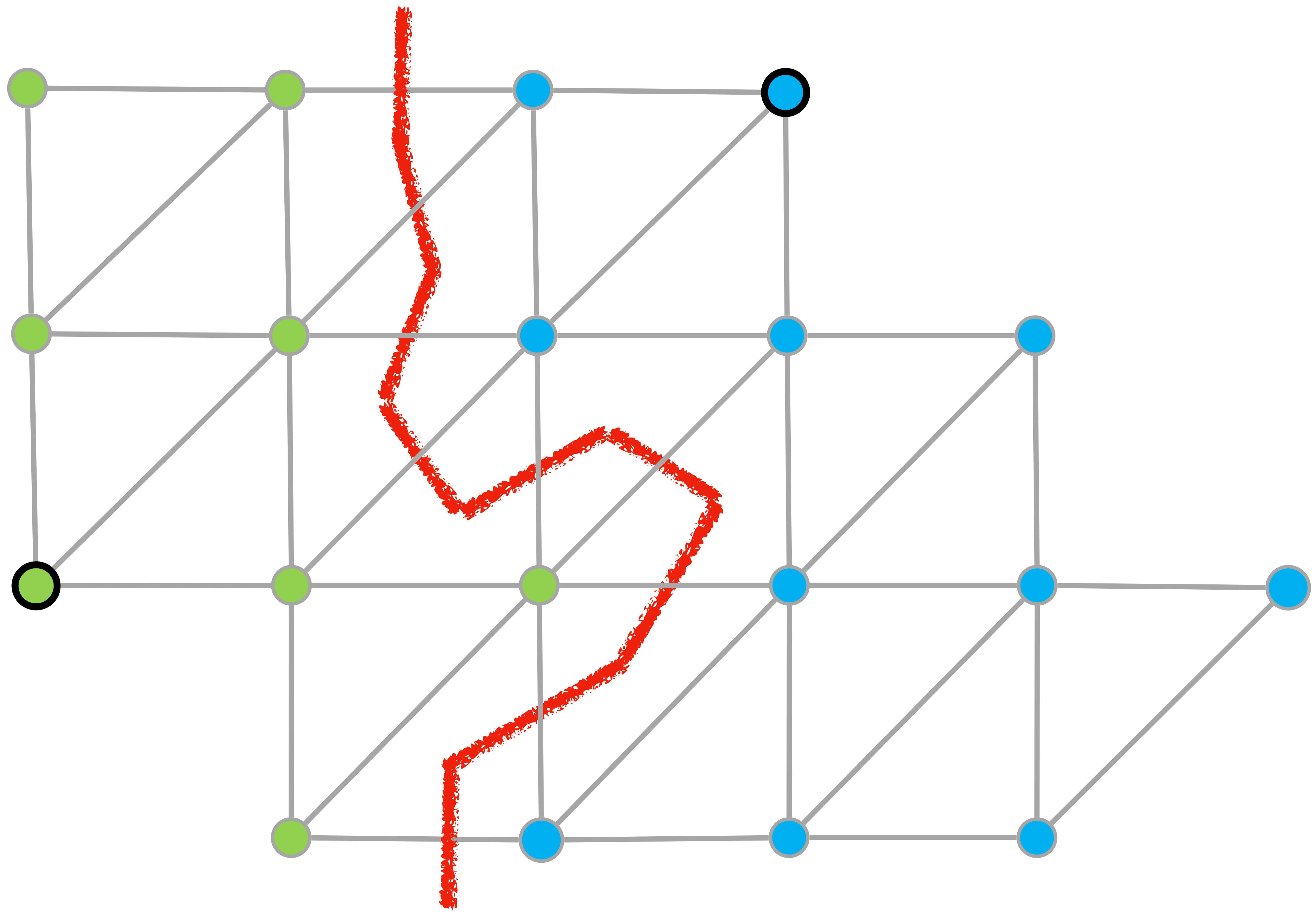
Voronoi vertices - adjacent to three different Voronoi cells (= **trichromatic faces**)



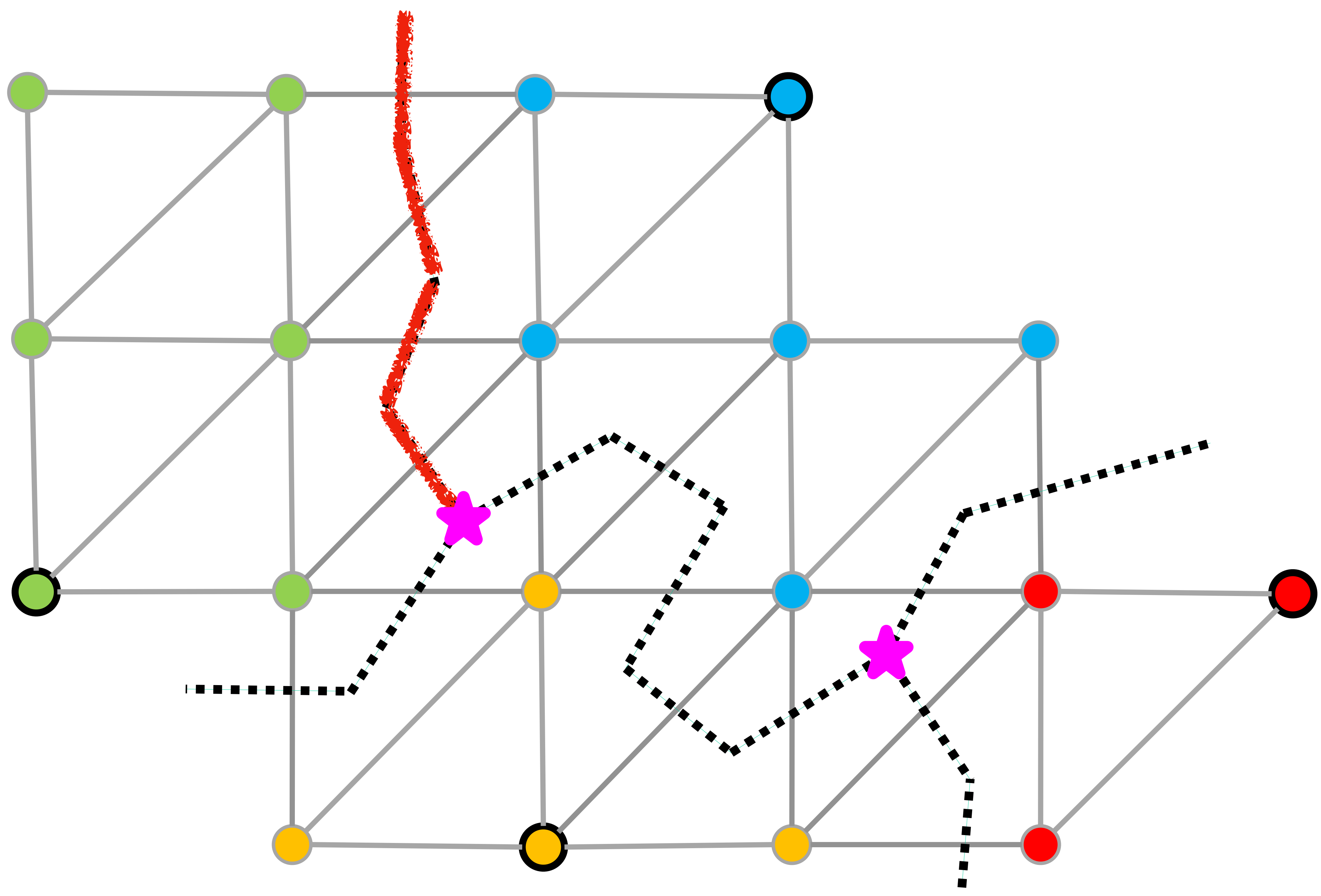
Voronoi diagram with b sites has b cells, $O(b)$ Voronoi vertices, and $O(b)$ Voronoi edges (by Euler's formula).



Bisectors - Voronoi diagram with just two sites

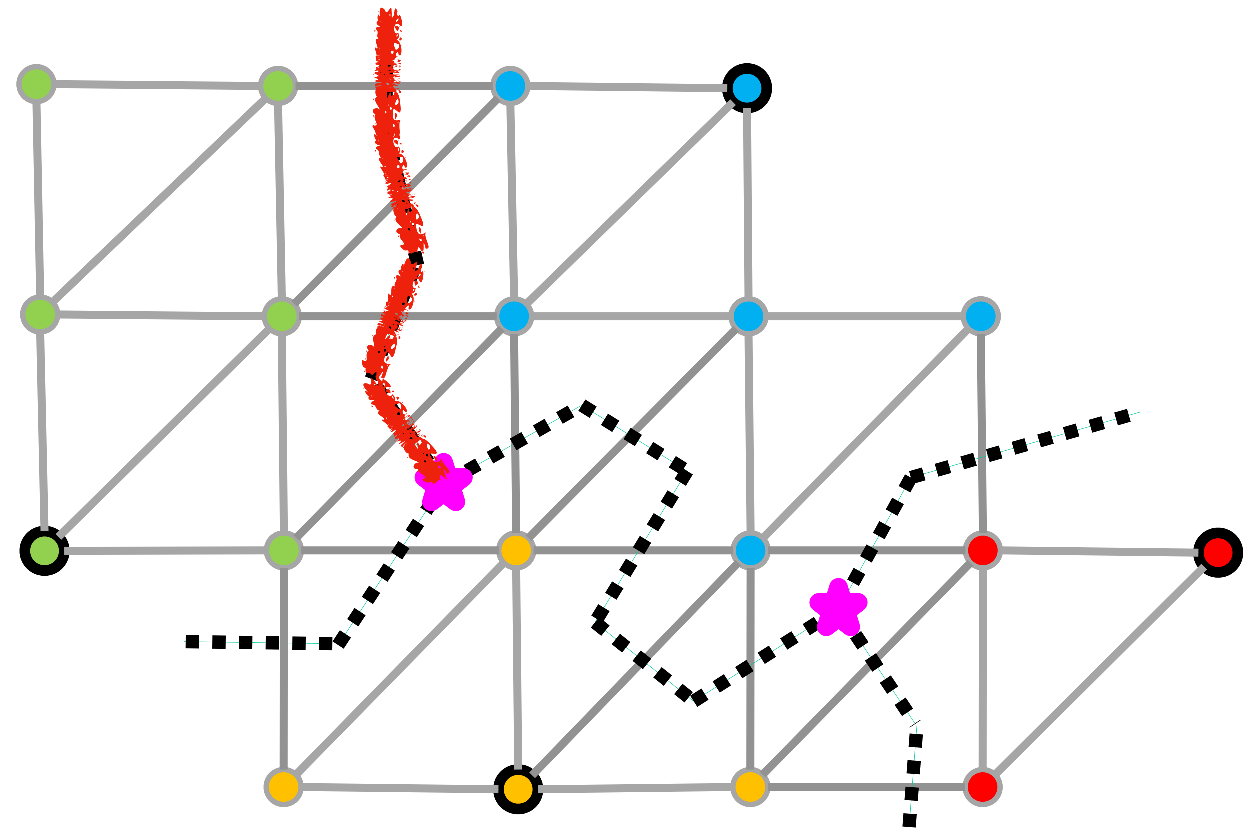


Every Voronoi edge is a subpath of a bisector

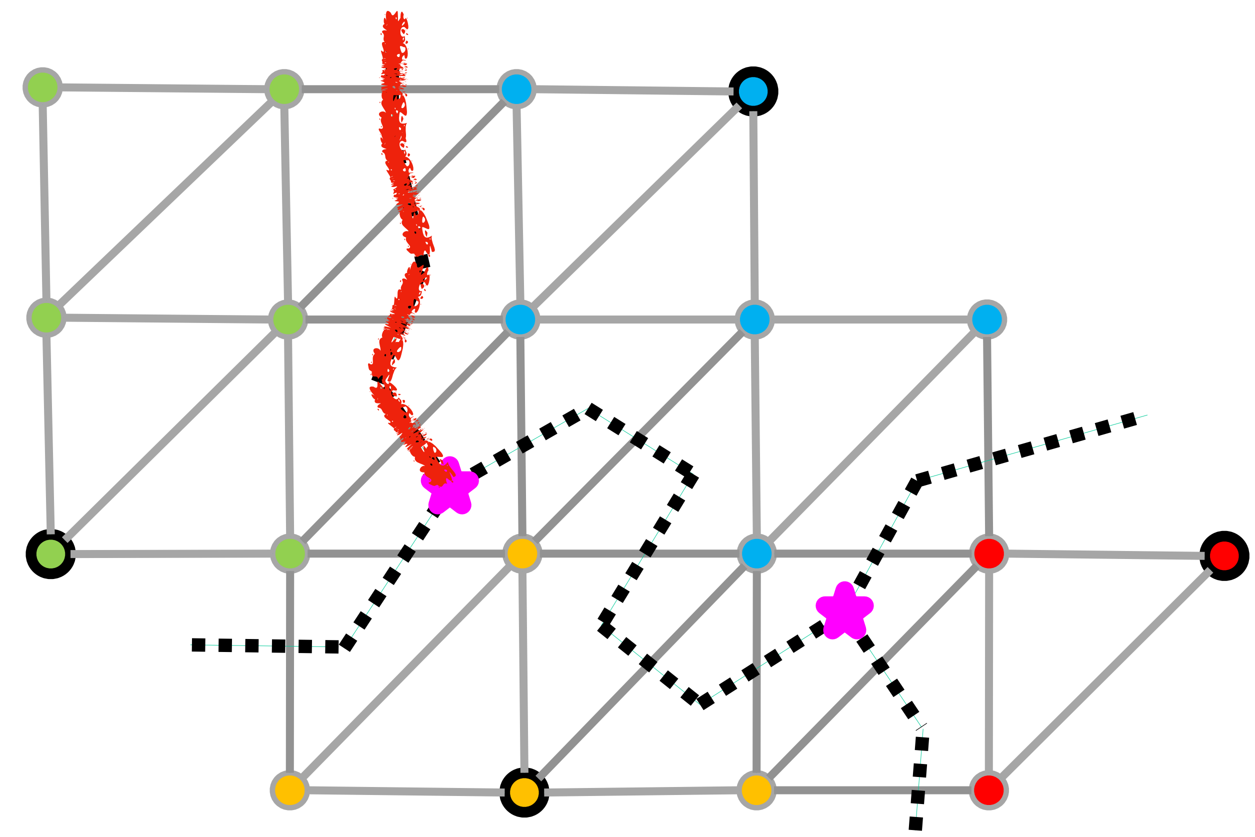


Strategy:

- Every Voronoi edge is a subpath of a bisector
- Every Voronoi vertex is the intersection of two bisectors



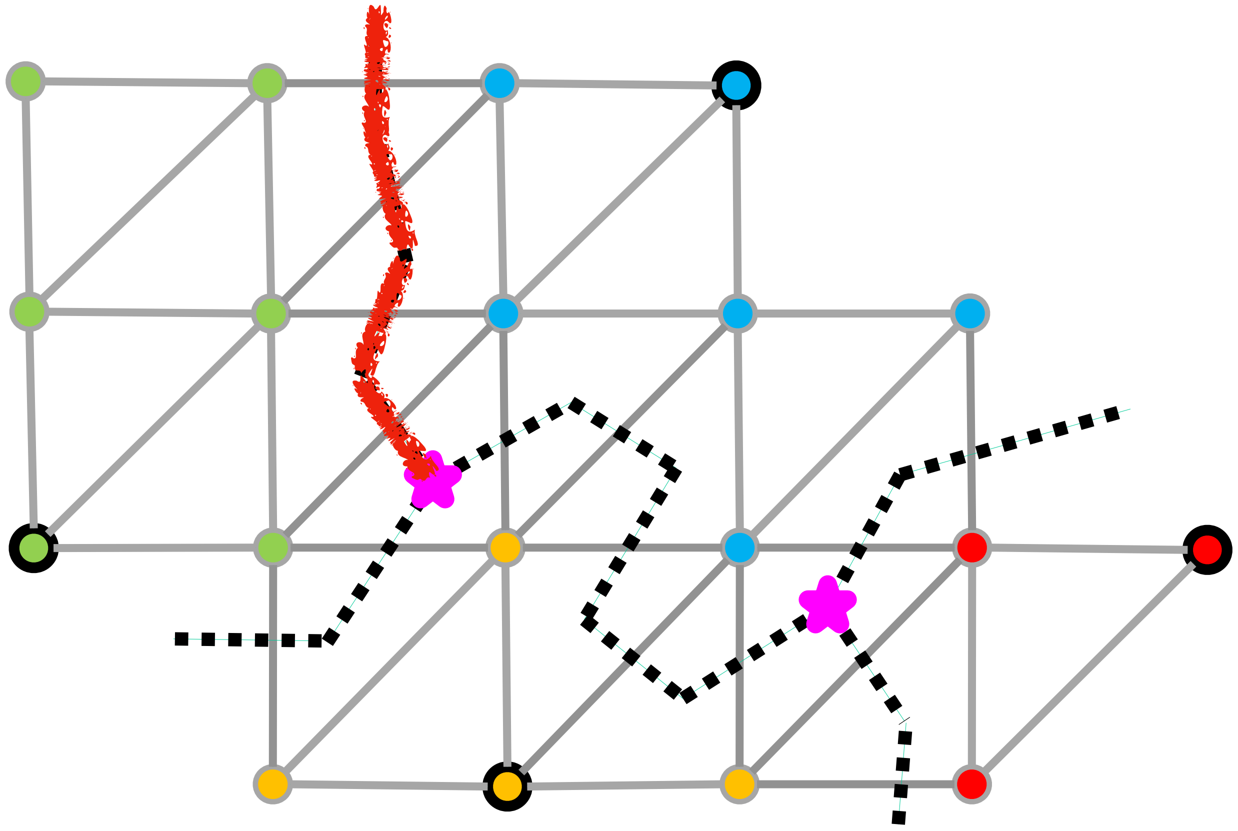
Strategy:



- Every Voronoi edge is a subpath of a bisector
- Every Voronoi vertex is the intersection of two bisectors

- Precompute and store all possible bisectors (for all pairs of sites and all possible weights…)

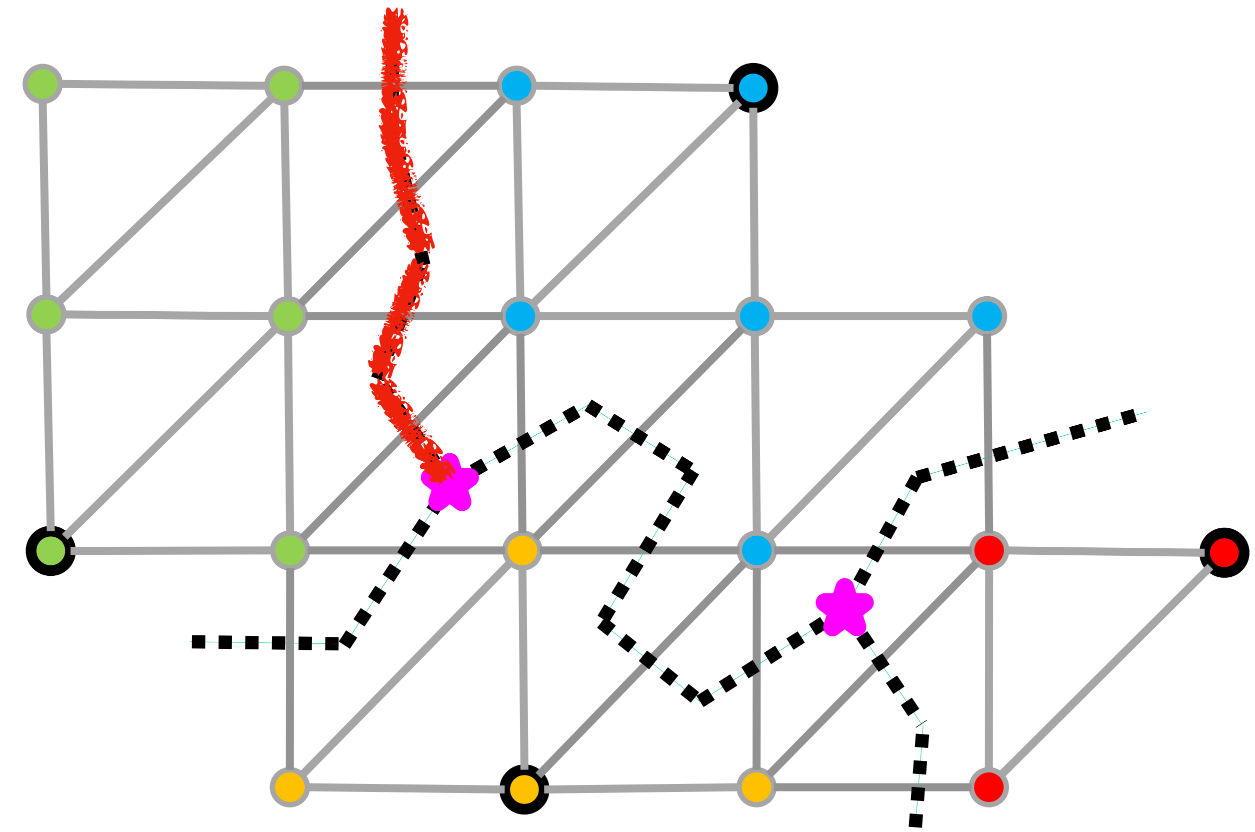
Strategy:



- Every Voronoi edge is a subpath of a bisector
- Every Voronoi vertex is the intersection of two bisectors

- Precompute and store all possible bisectors (for all pairs of sites and all possible weights…)
- Represent Voronoi edges as subpaths of bisectors

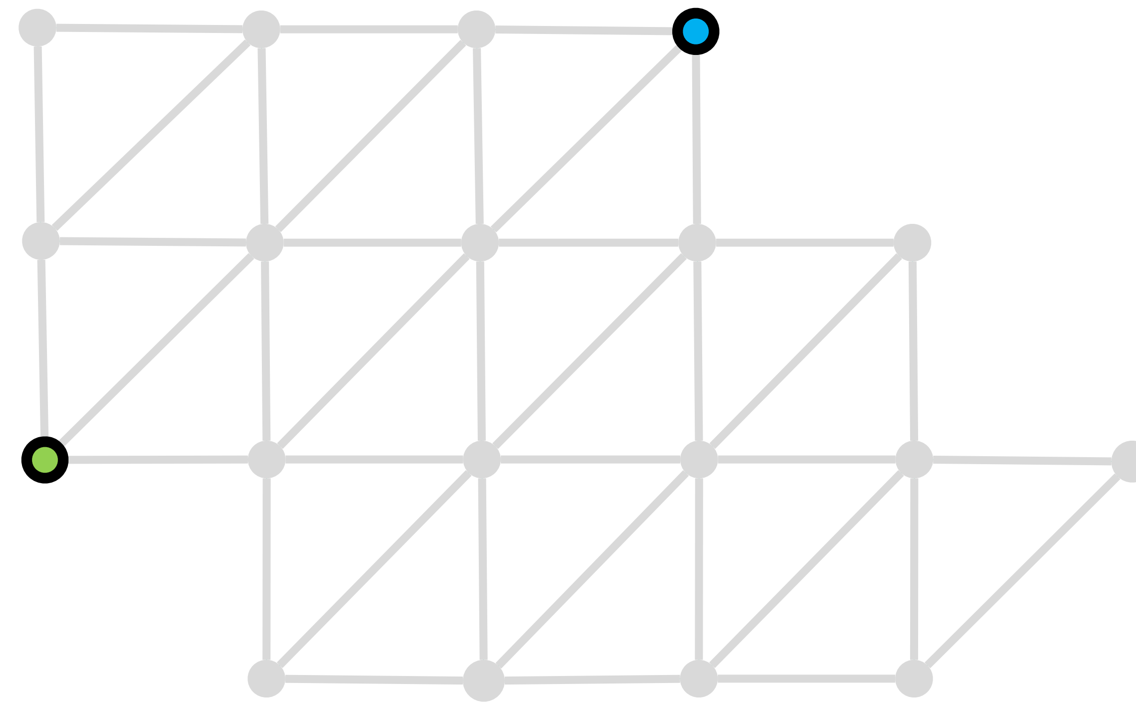
Strategy:



- Every Voronoi edge is a subpath of a bisector
- Every Voronoi vertex is the intersection of two bisectors

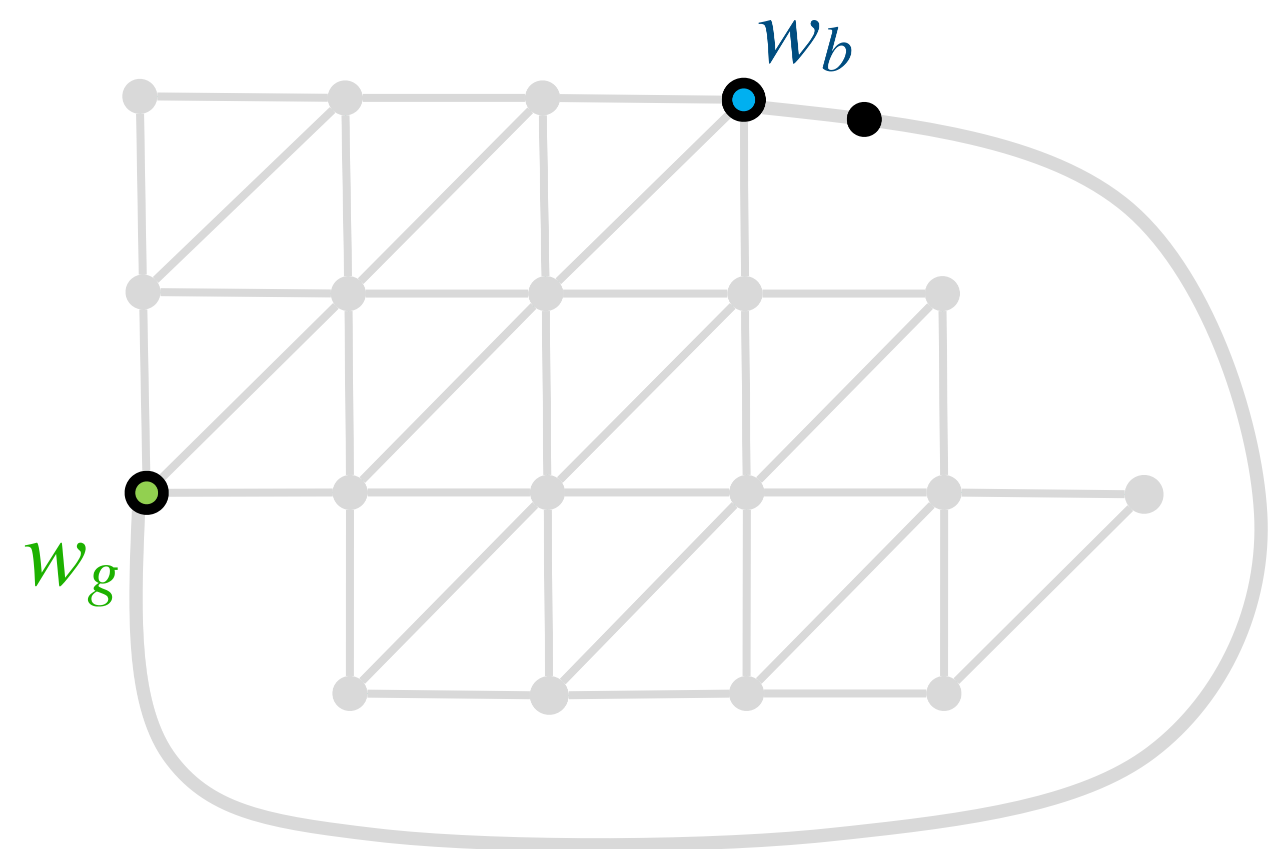
- Precompute and store all possible bisectors (for all pairs of sites and all possible weights…)
- Represent Voronoi edges as subpaths of bisectors
- Construct Voronoi diagram with b sites in $\tilde{O}(b)$ time using divide and conquer by intersecting bisectors

Computing bisectors



Computing bisectors

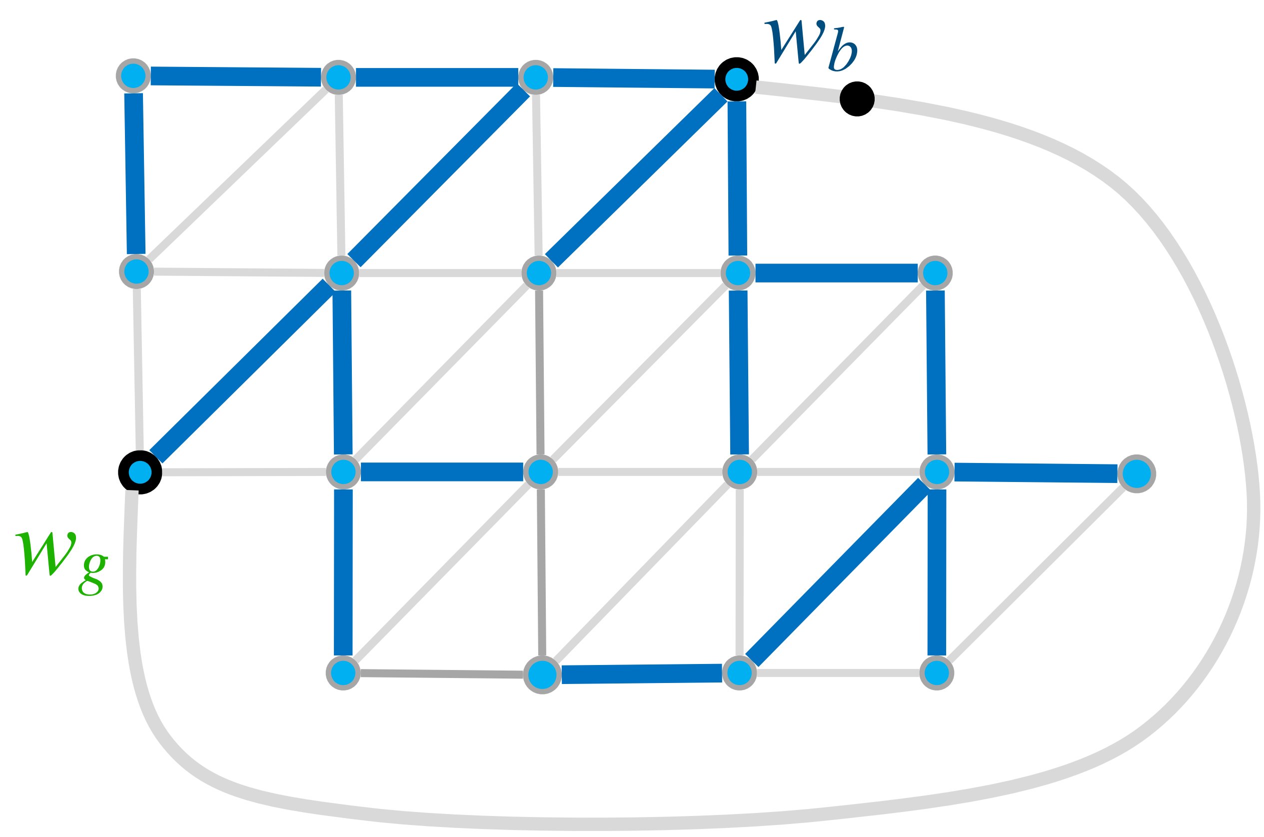
- only depends on the difference w_b-w_g



reminiscent of MSSP
[Cabello, Chambers, Erickson]

Computing bisectors

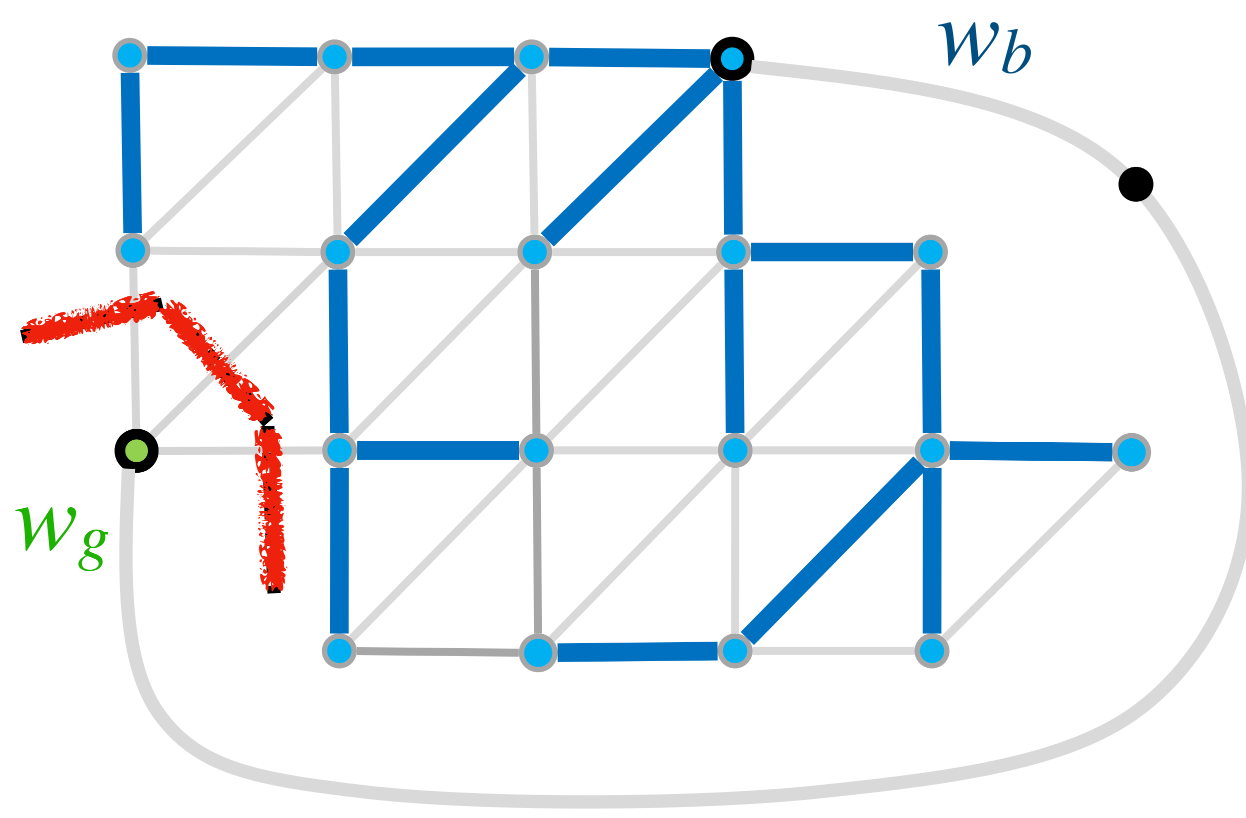
- only depends on the difference w_b-w_g
- as we increase w_b-w_g the bisector sweeps the graph
- changes occur at discrete critical values, where blue vertices become green



reminiscent of MSSP
[Cabello, Chambers, Erickson]

Computing bisectors

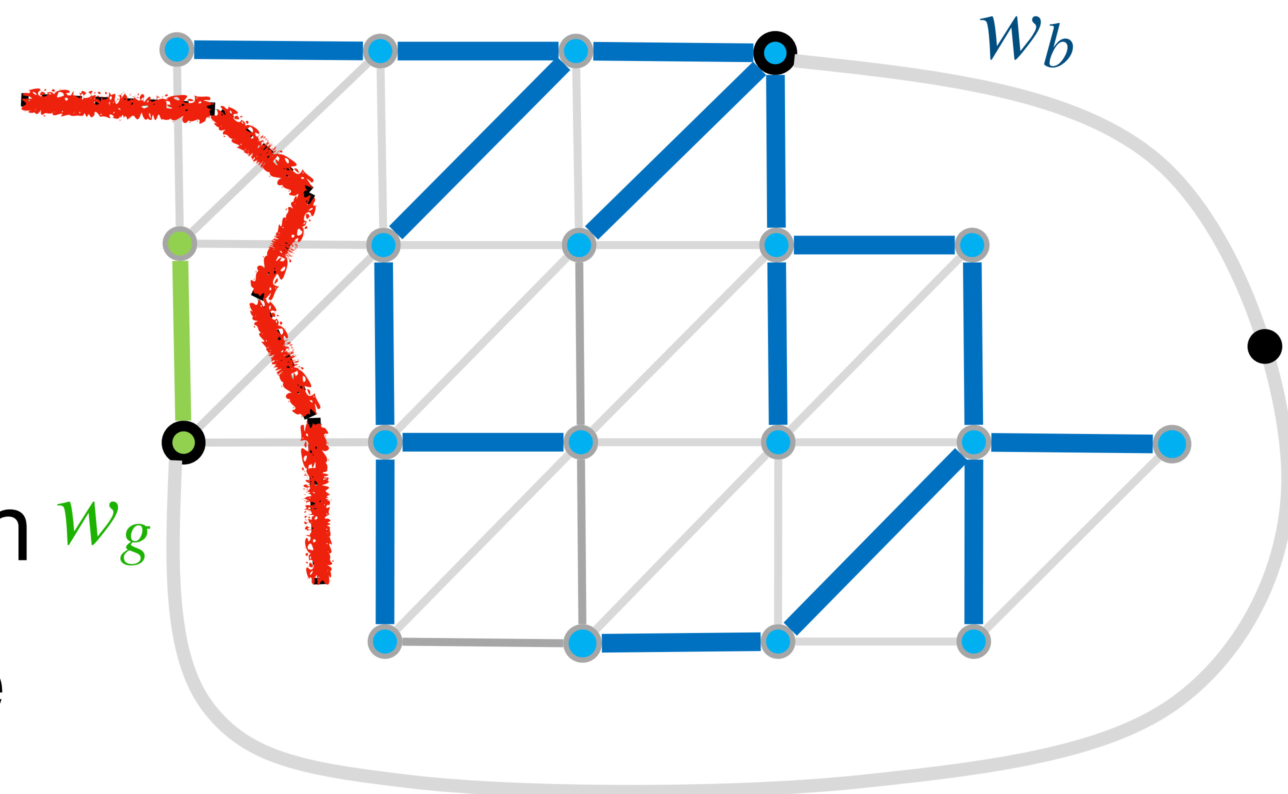
- only depends on the difference w_b-w_g
- as we increase w_b-w_g the bisector sweeps the graph
- changes occur at discrete critical values, where blue vertices become green



reminiscent of MSSP
[Cabello, Chambers, Erickson]

Computing bisectors

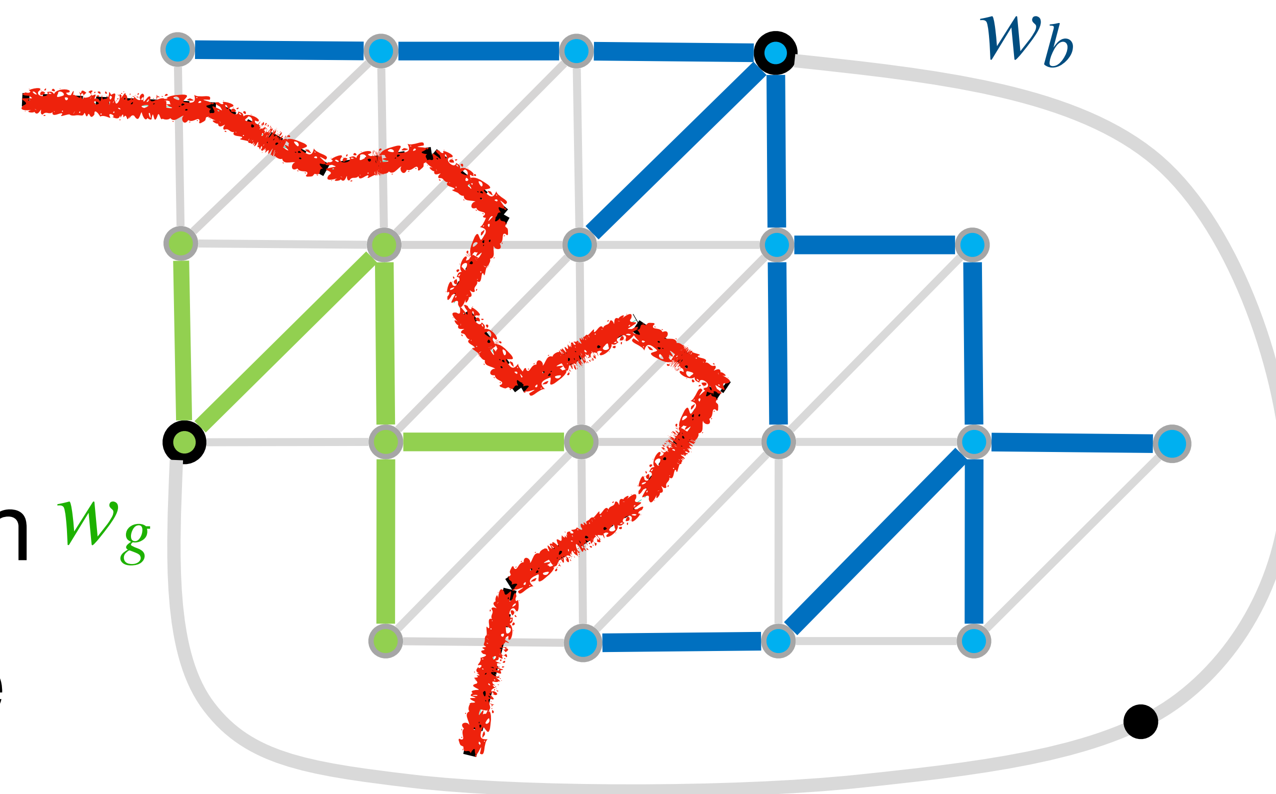
- only depends on the difference w_b-w_g
- as we increase w_b-w_g the bisector sweeps the graph
- changes occur at discrete critical values, where blue vertices become green



reminiscent of MSSP
[Cabello, Chambers, Erickson]

Computing bisectors

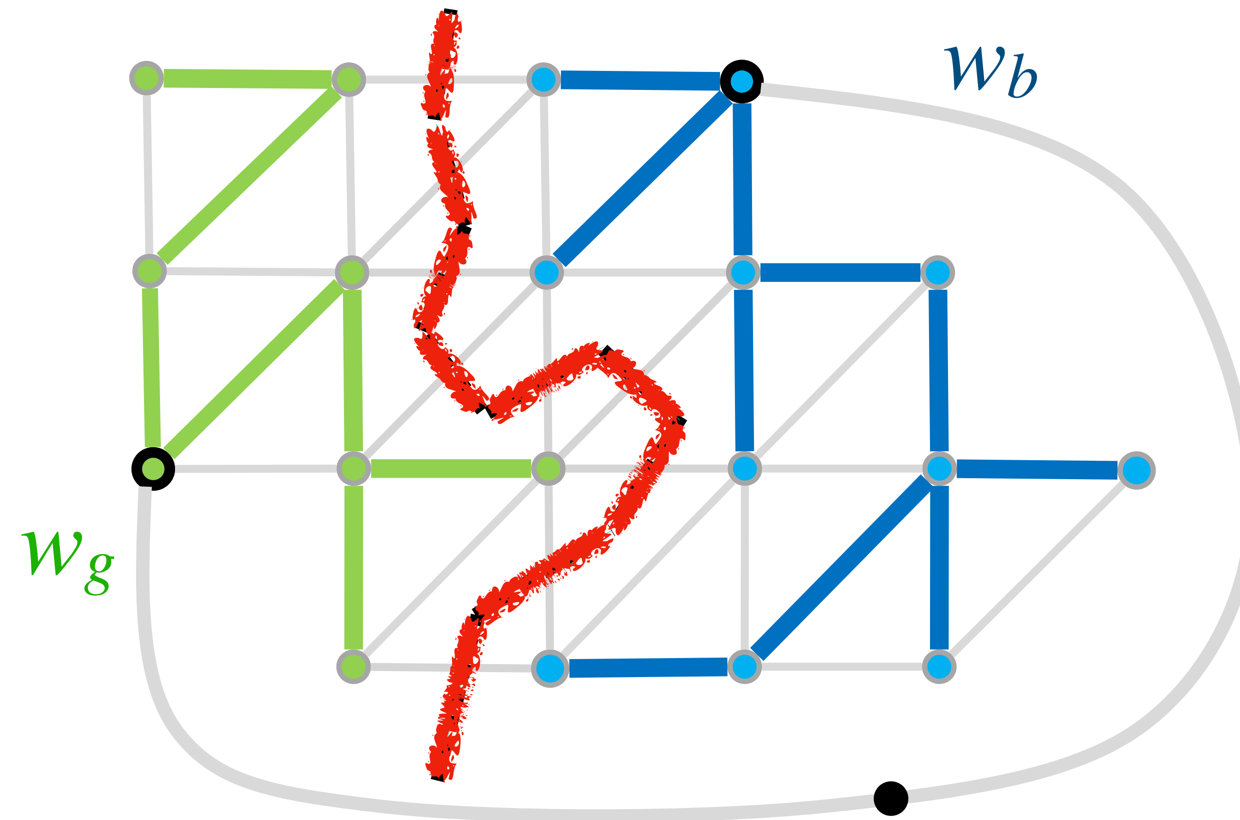
- only depends on the difference w_b-w_g
- as we increase w_b-w_g the bisector sweeps the graph
- changes occur at discrete critical values, where blue vertices become green



reminiscent of MSSP
[Cabello, Chambers, Erickson]

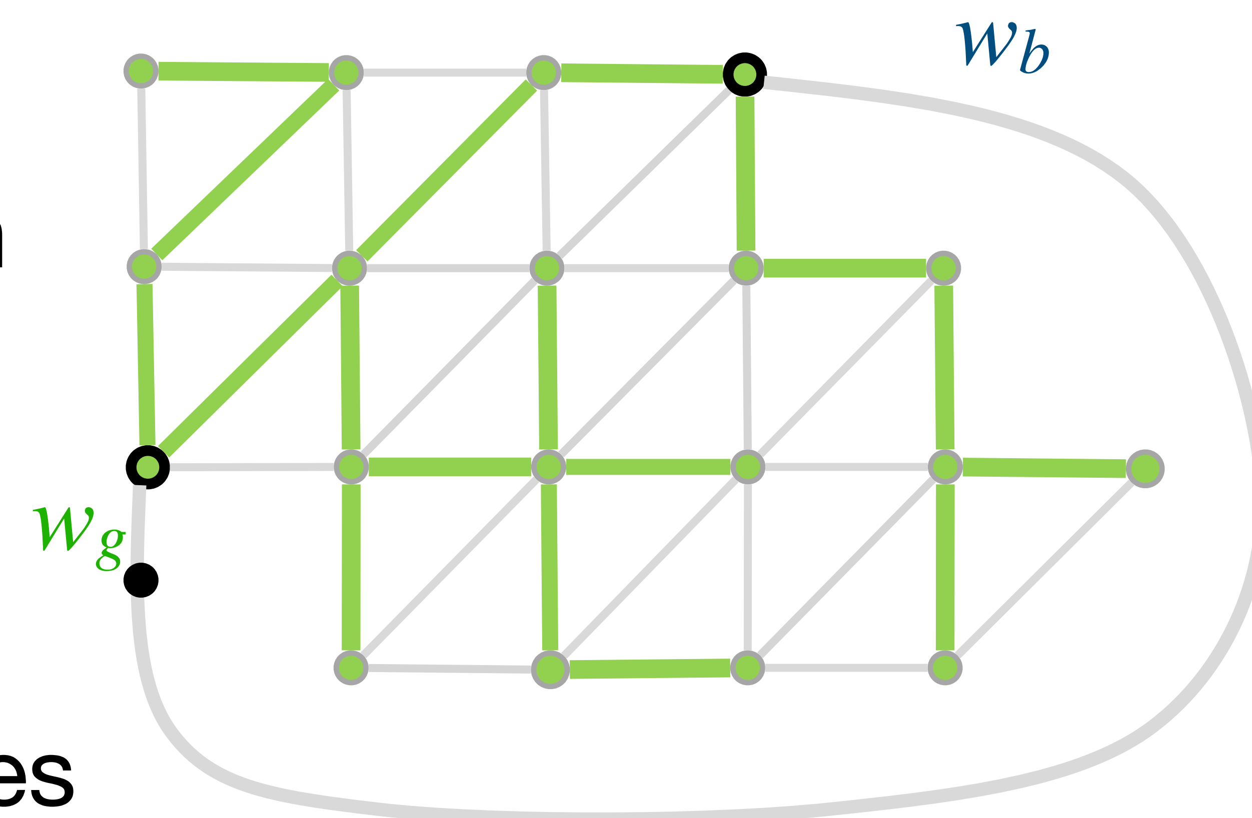
Computing bisectors

- only depends on the difference w_b-w_g
- as we increase w_b-w_g the bisector sweeps the graph
- changes occur at discrete critical values, where blue vertices become green



reminiscent of MSSP
[Cabello, Chambers, Erickson]

Computing bisectors



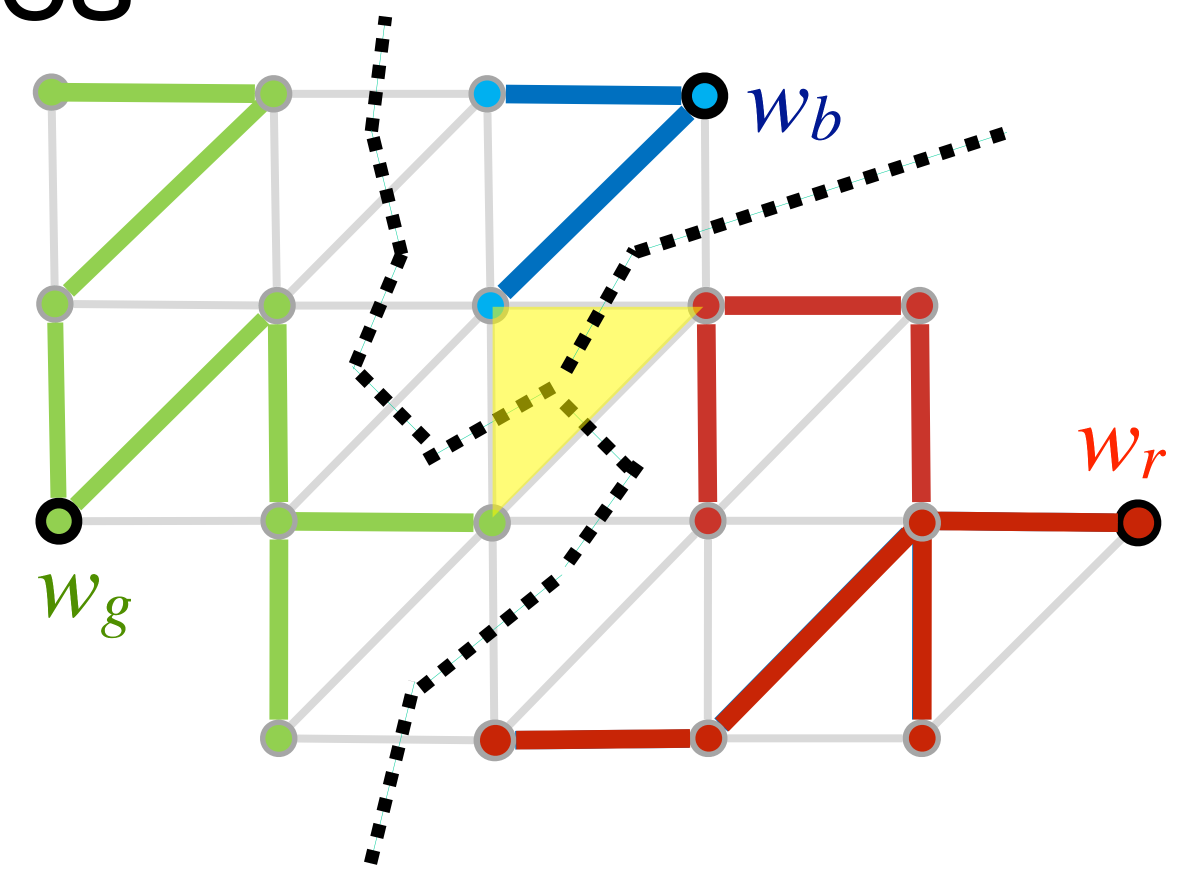
- as we increase w_b-w_g the bisector sweeps the graph
- changes occur at discrete critical values, where blue vertices become green

➡ In a graph with $O(r)$ vertices there are only $O(r)$ bisectors (for each pair of sites). can be computed in $\tilde{O}(r)$ time

- for each pair of sites, all bisectors stored in $\tilde{O}(r)$ space and time using persistent binary search trees
- for $r^{1/2}$ sites total preprocessing $\tilde{O}(r^2)$ time and space
- compared to $\tilde{O}(r^3)$ time and space in Cabello's

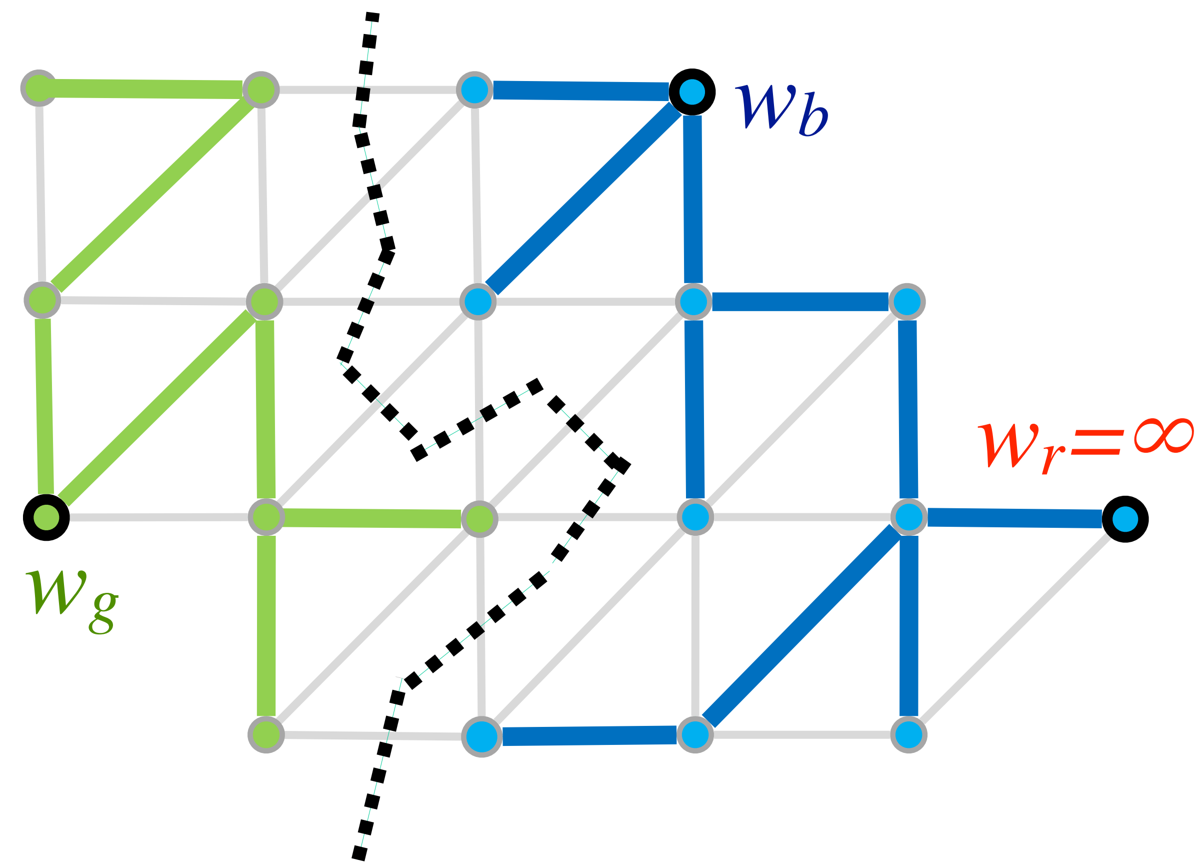
Intersecting bisectors = finding trichromatic faces

- three sites with weights w_b, w_g, w_g.
- want to find a face with one vertex in each of the Voronoi cells.



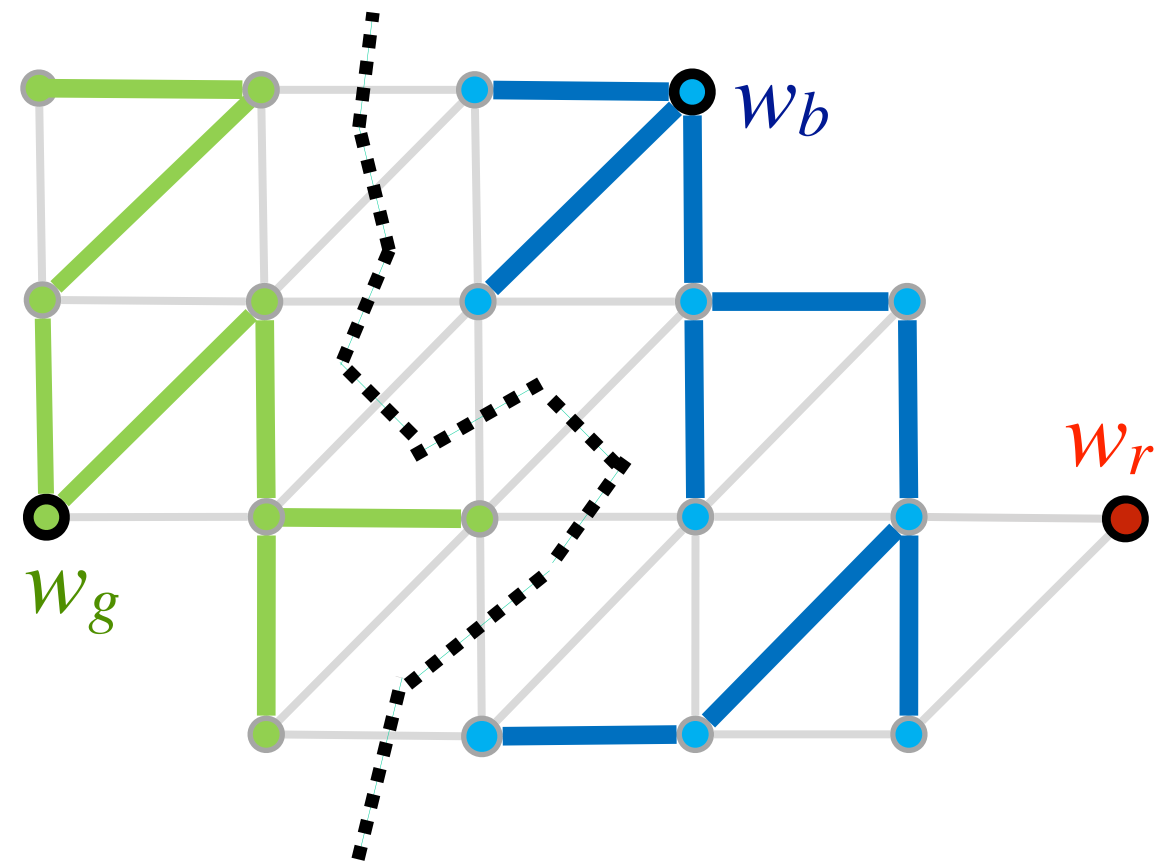
Dynamics of trichromatic faces

- fix w_b and w_g, and gradually decrease w_g.



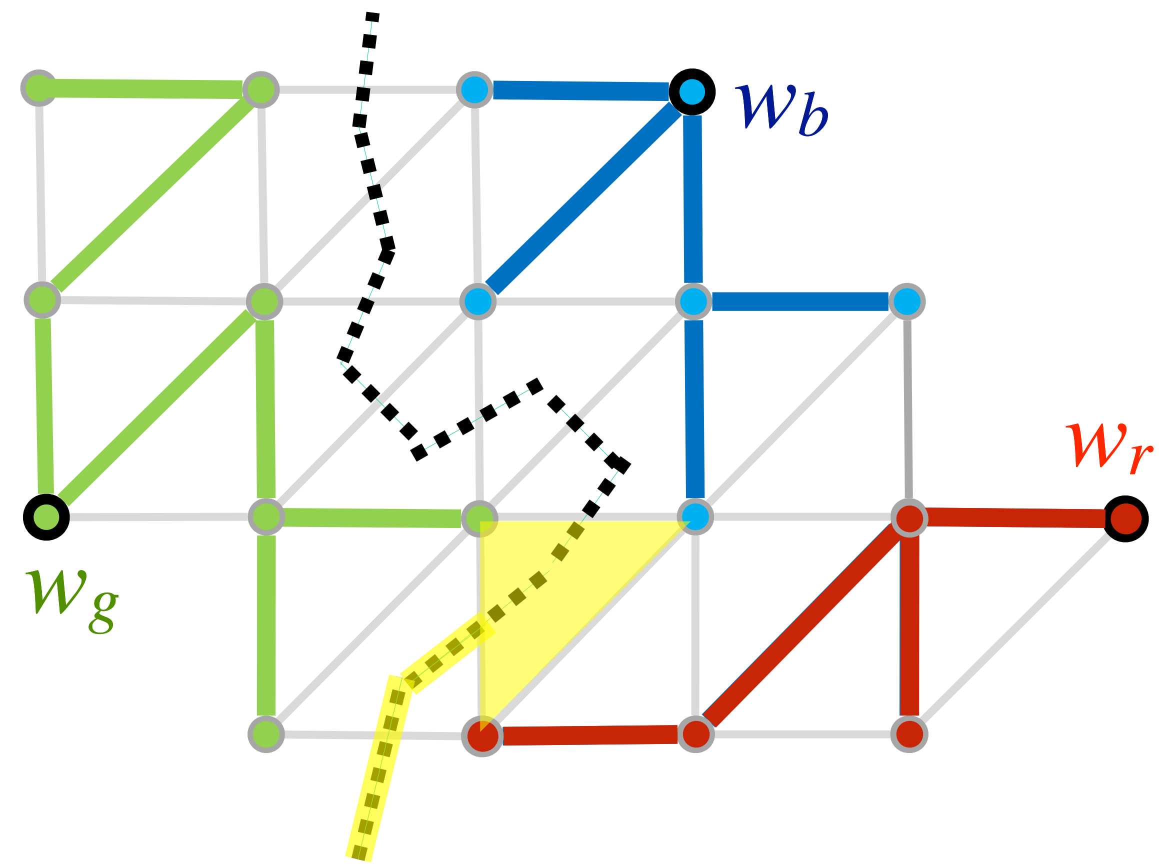
Dynamics of trichromatic faces

- fix w_b and w_g, and gradually decrease w_g.



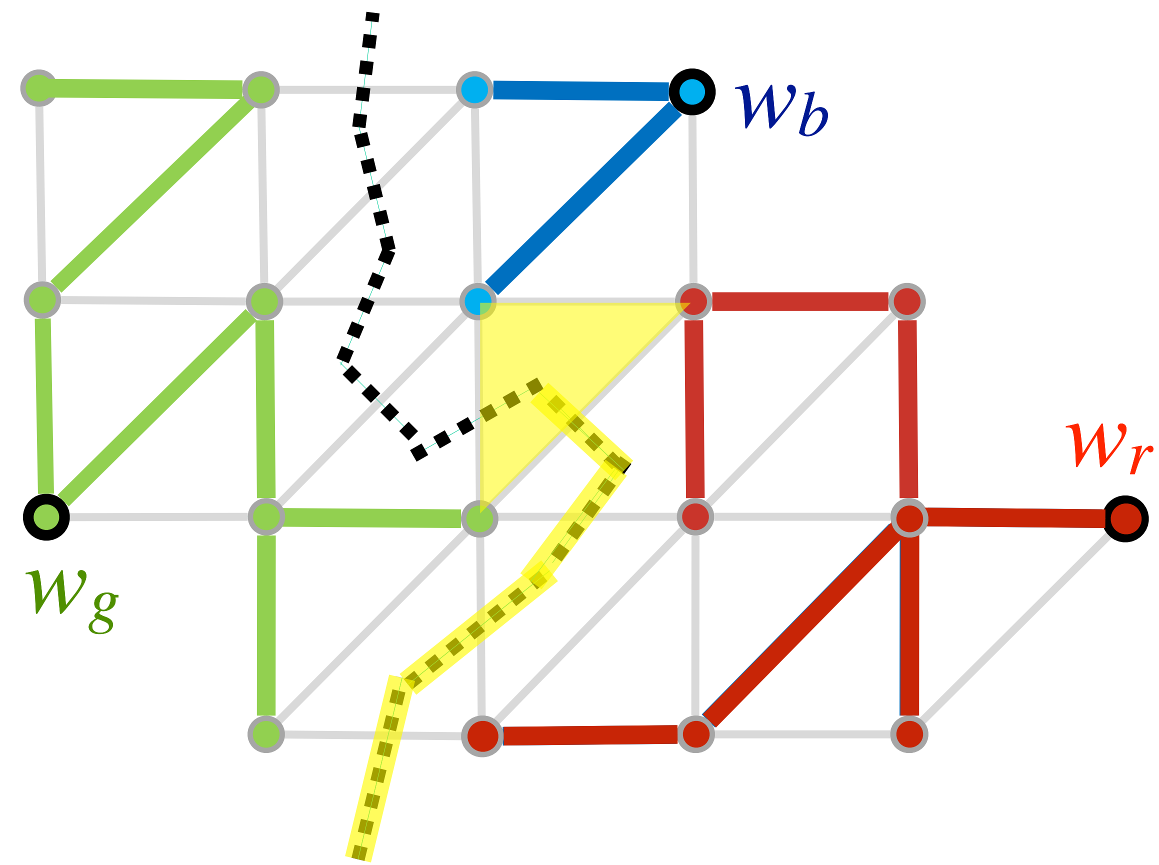
Dynamics of trichromatic faces

- fix w_b and w_g, and gradually decrease w_g.
- as we decrease w_r, the trichromatic face moves monotonically along the (g,b)-bisector.



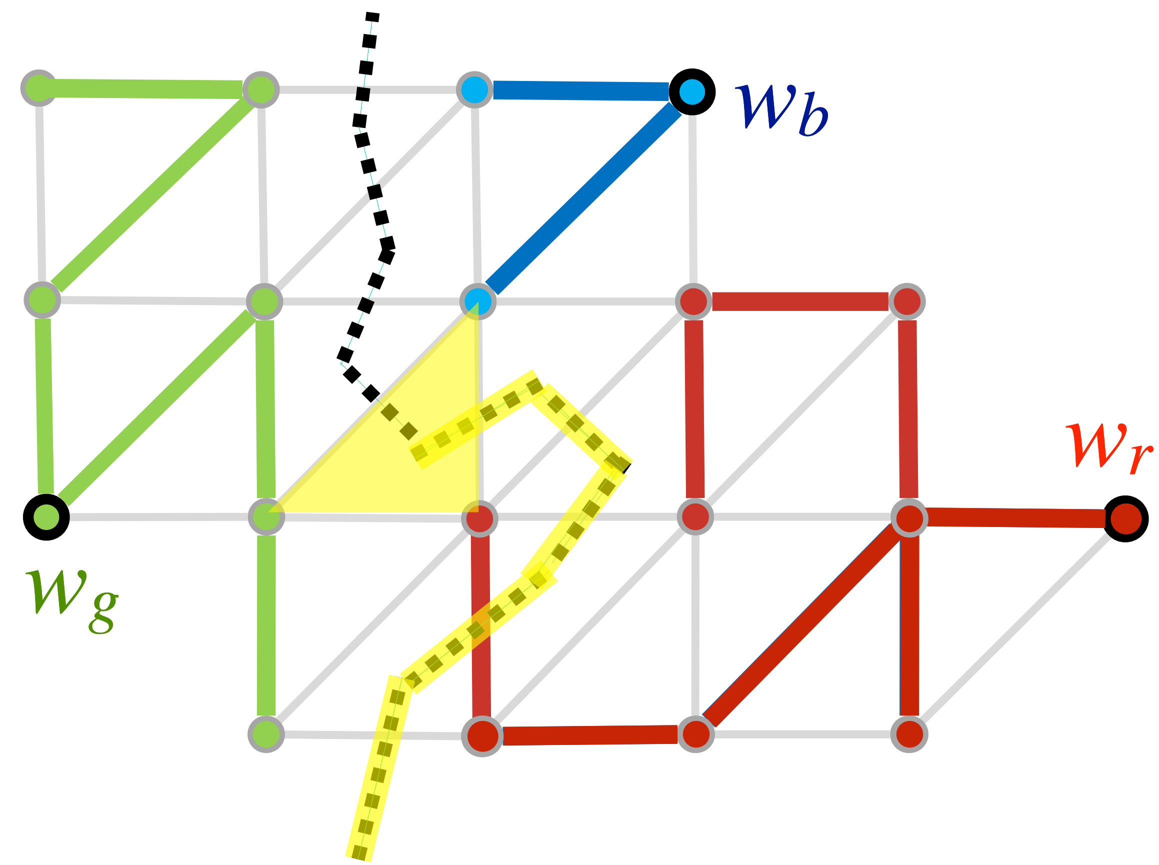
Dynamics of trichromatic faces

- fix w_b and w_g, and gradually decrease w_g.
- as we decrease w_r, the trichromatic face moves monotonically along the (g,b)-bisector.



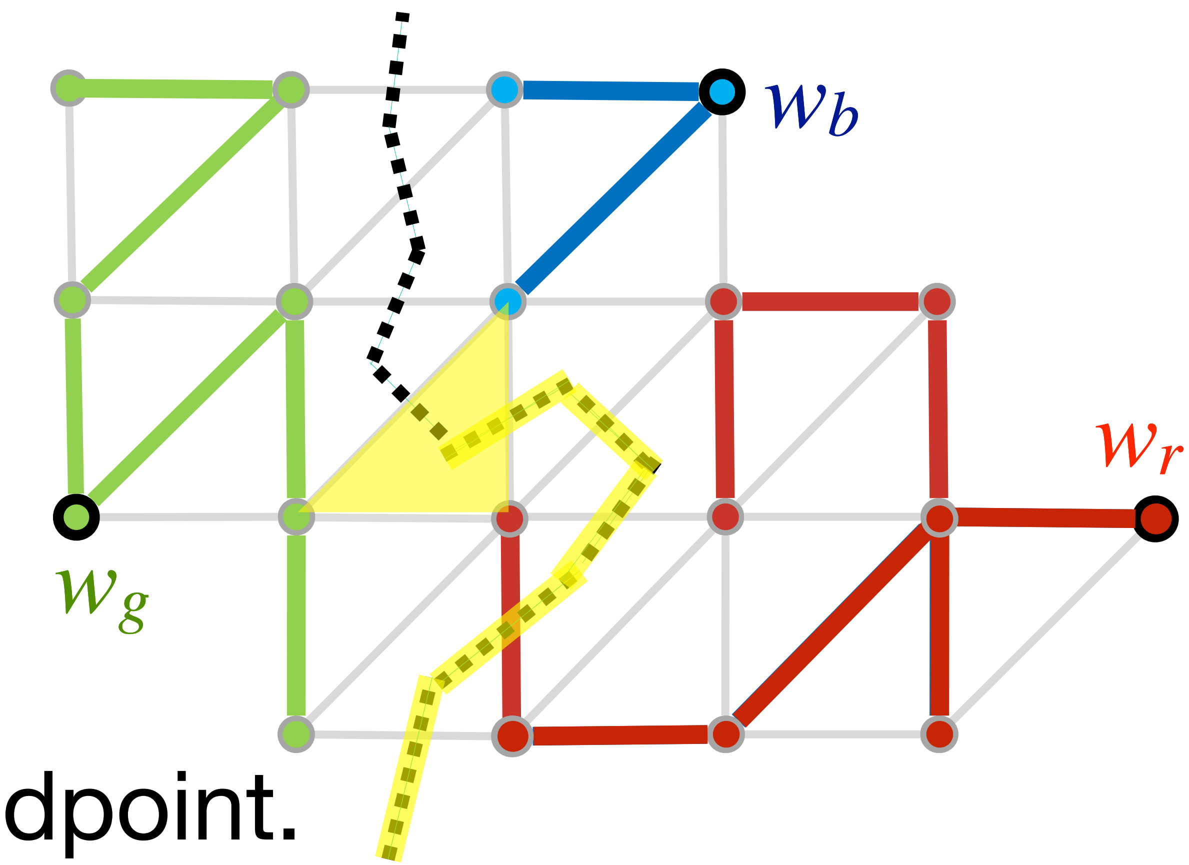
Dynamics of trichromatic faces

- fix w_b and w_g, and gradually decrease w_g.
- as we decrease w_r, the trichromatic face moves monotonically along the (g,b)-bisector.



Finding trichromatic faces

- as we decrease w_r, the trichromatic face moves monotonically along the (g,b)-bisector.
- given w_b, w_g, w_r can determine in constant time whether an edge has a red endpoint.

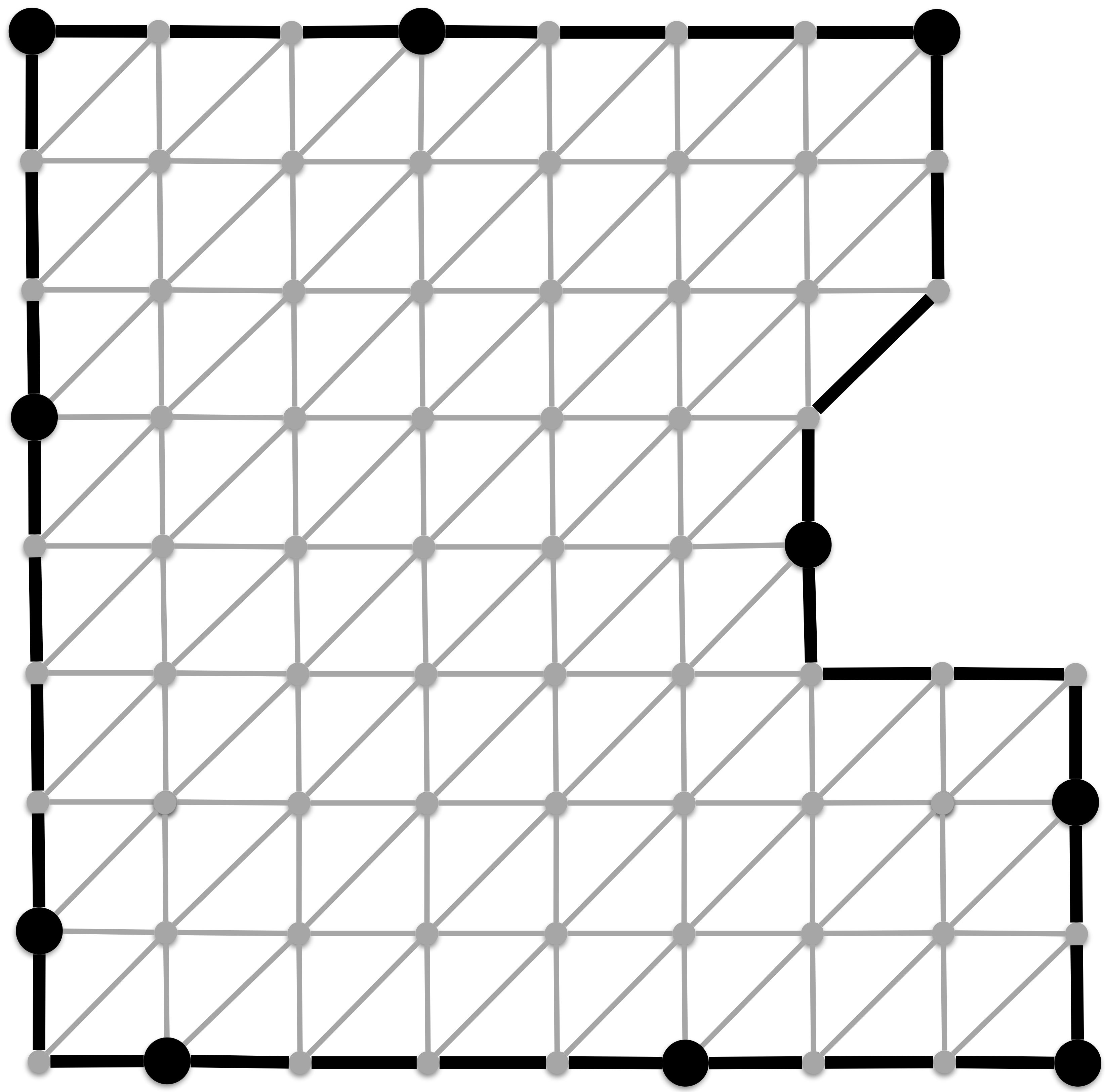


- binary search for the last edge on the (g,b)-bisector that has a red endpoint
- takes $\tilde{O}(1)$ time
- need to extend to groups of sites. Becomes much more complicated

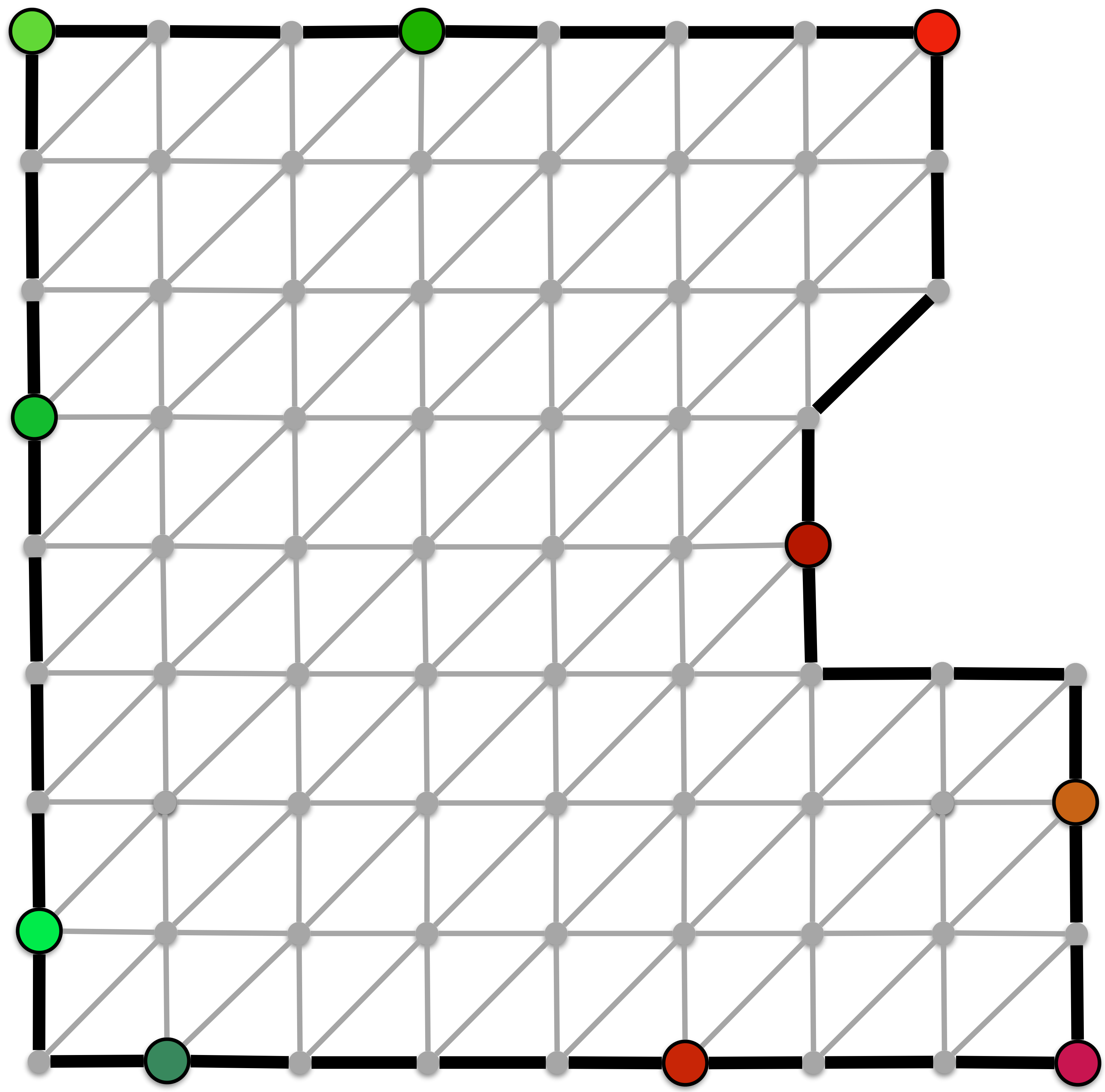
Constructing a Voronoi diagram

- we have precomputed bisectors in $\tilde{O}(r^2)$
- we know how to find a trichromatic face in $\tilde{O}(1)$ time
- we compute the weighted Voronoi diagram of $r^{1/2}$ sites in $\tilde{O}(r^{1/2})$ time using divide and conquer.

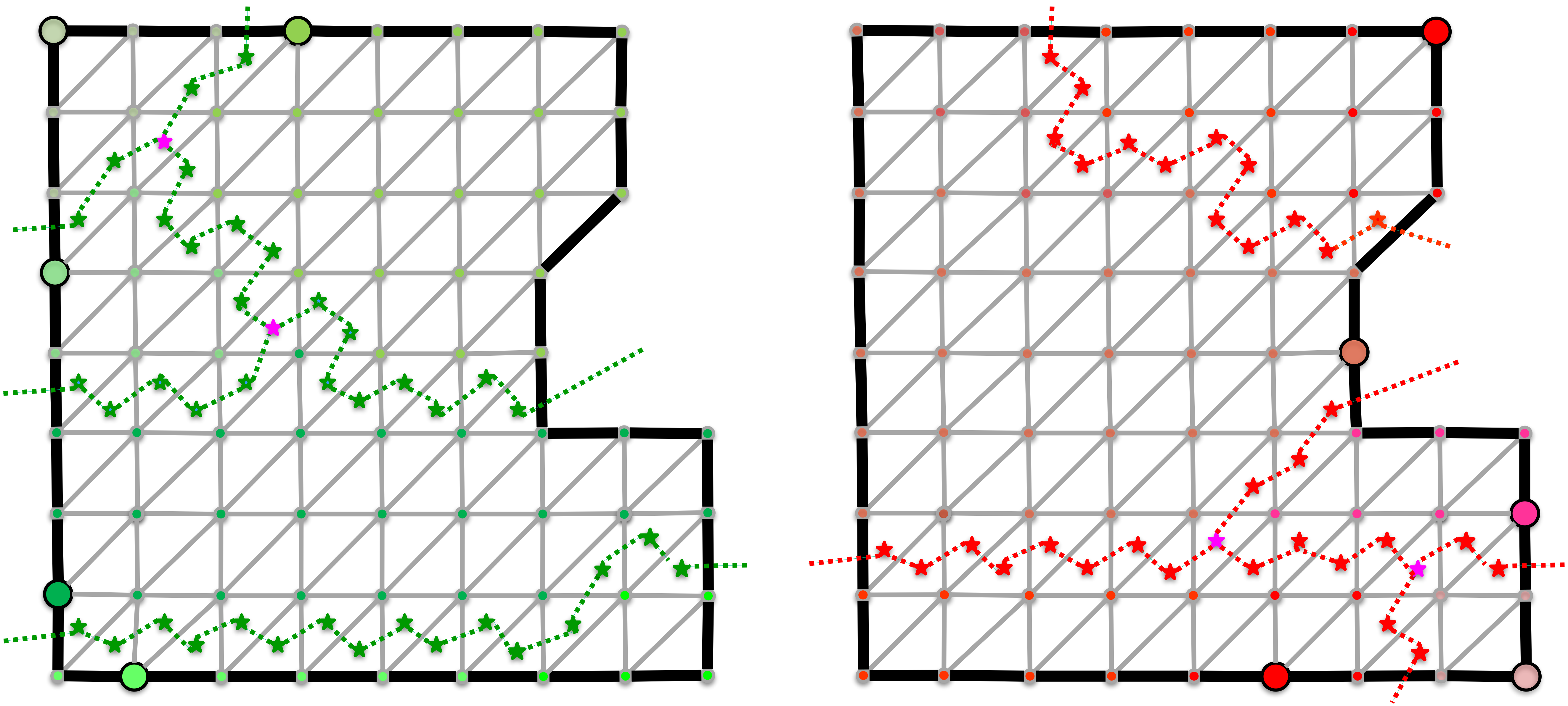
Constructing diagram via divide and conquer

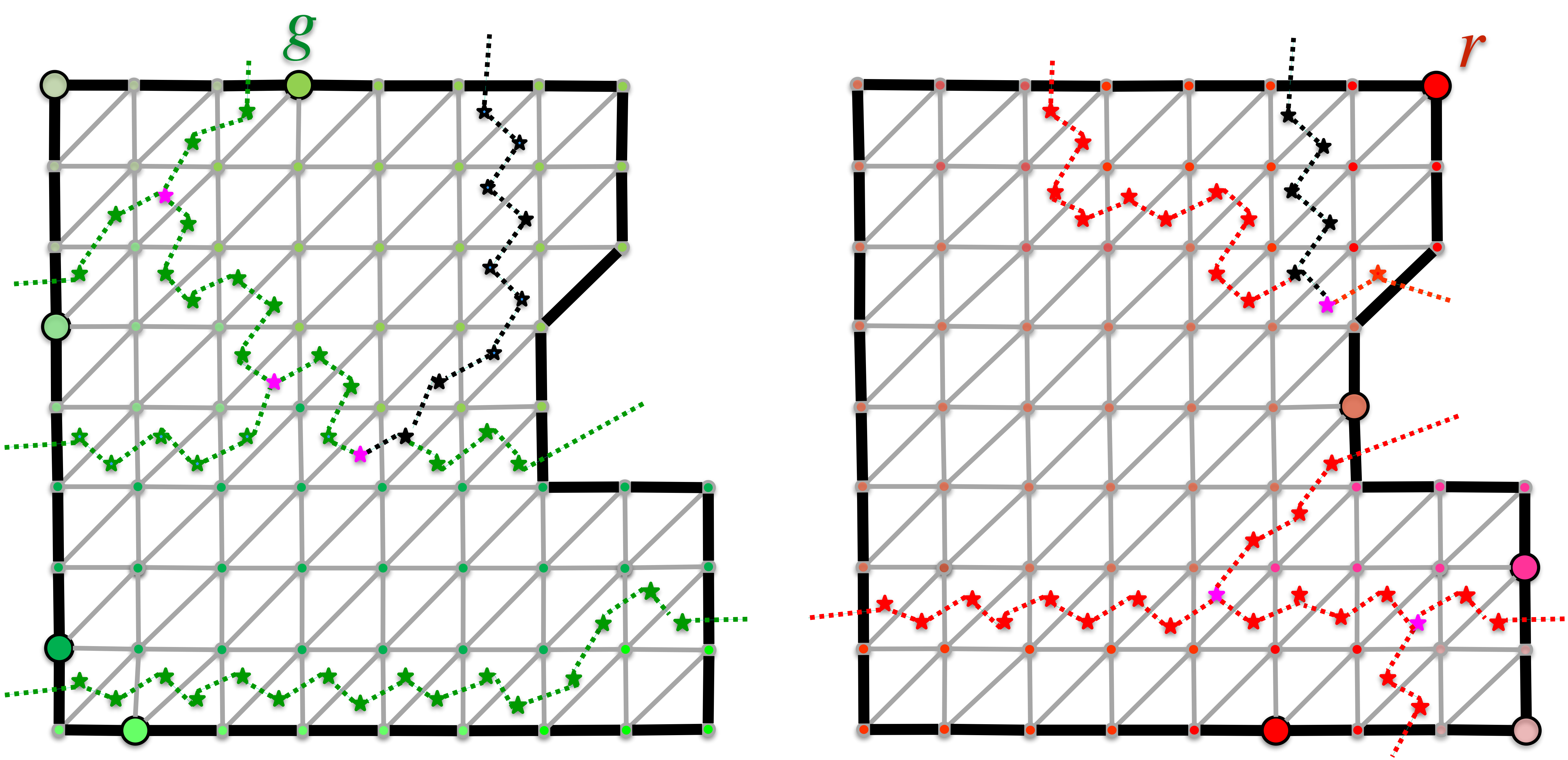


Constructing diagram via divide and conquer

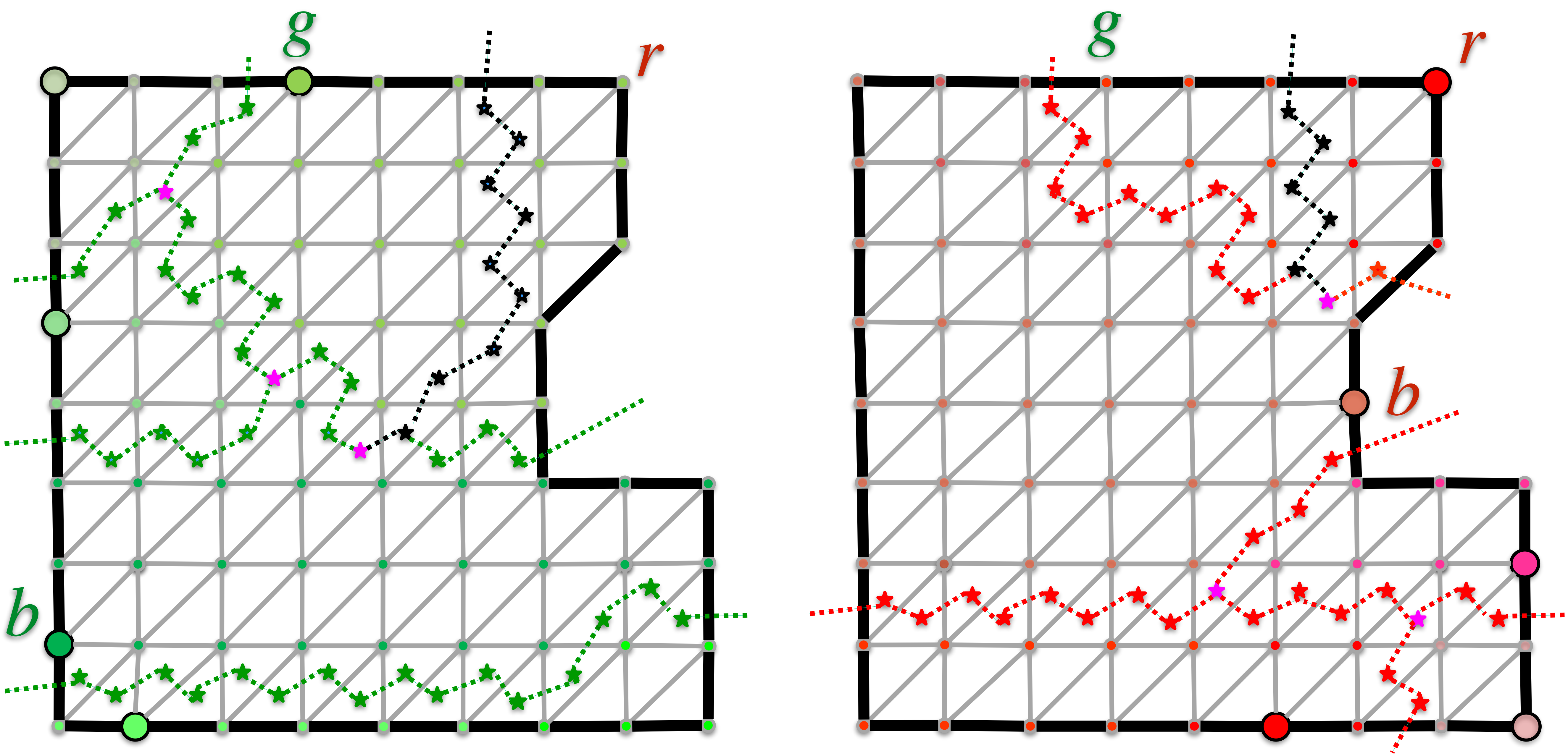


Constructing diagram via divide and conquer



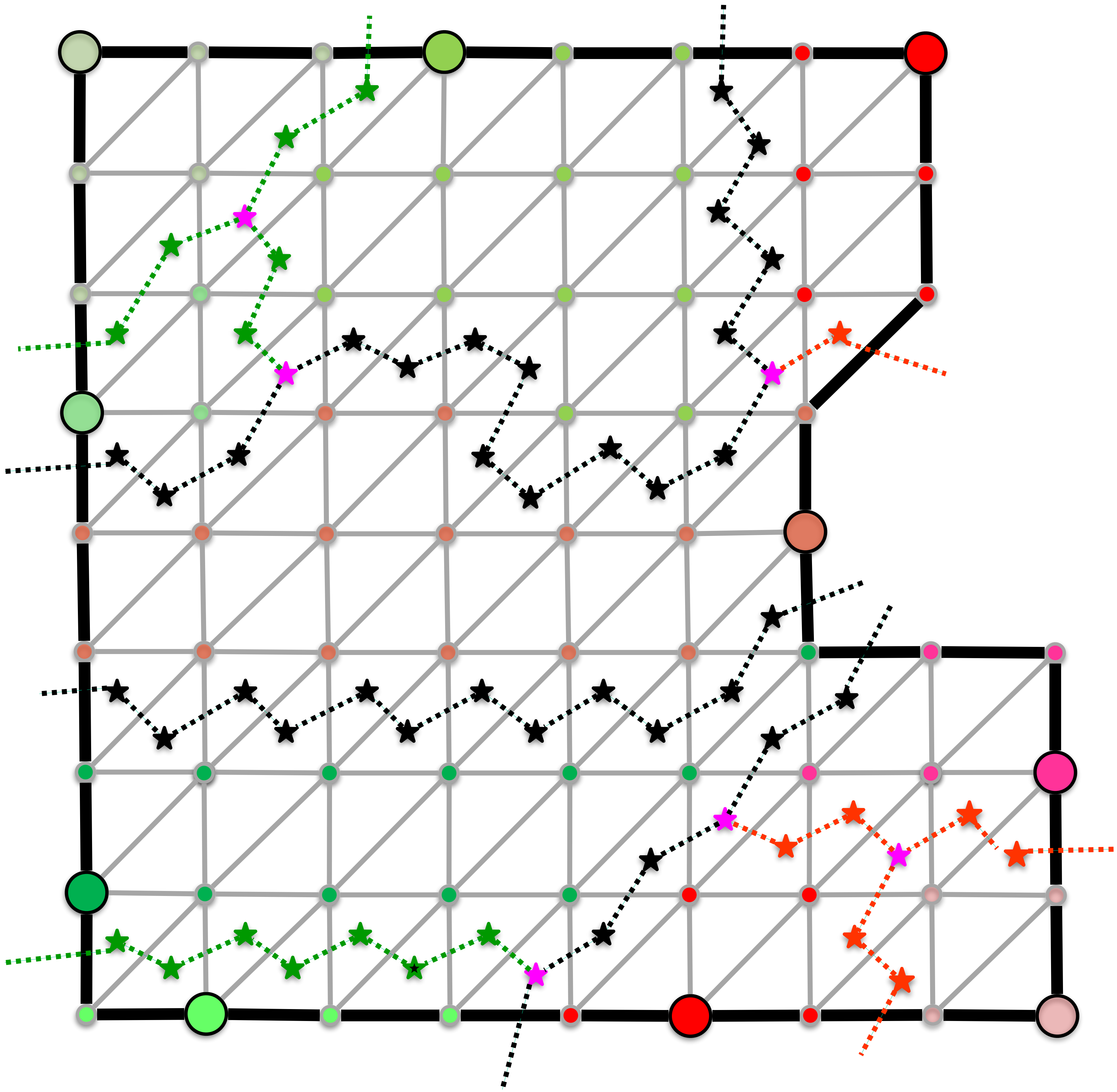


g
r



g
r
b
g
r
b

Combined diagram



Constructing a Voronoi diagram

- since we can compute trichromatic vertices in $\tilde{O}(1)$ time, we can merge two Voronoi diagrams with b_1 and b_2 sites in $\tilde{O}(b_1 + b_2)$ time
- so constructing a Voronoi diagram with $r^{1/2}$ sites takes $\tilde{O}(r^{1/2})$ time

Things I swept under the rug

- handling sites on more than one face (holes)
- mechanism for finding furthest vertex in a Voronoi cell (similar to Cabello's)

Conclusion

- efficient deterministic construction of Voronoi diagrams on planar graphs
- diameter of a planar graph in deterministic $\tilde{O}(n^{5/3})$ time

- can we get below for $\tilde{O}(n^{5/3})$ time diameter?
- nearly linear time / (conditional) lower bounds?
- what other problems can benefit from these techniques?