Improved Bounds for Discrete Voronoi games

Mark de Berg and Geert van Wordragen

EINDHOVEN UNIVERSITY OF TECHNOLOGY

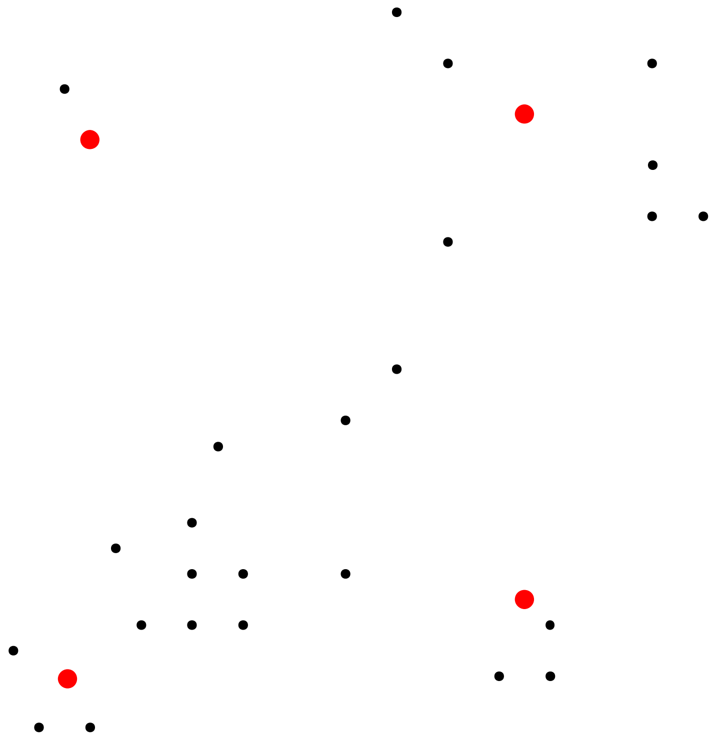
Aalto University

The setting

- Voters and parties are 2D points

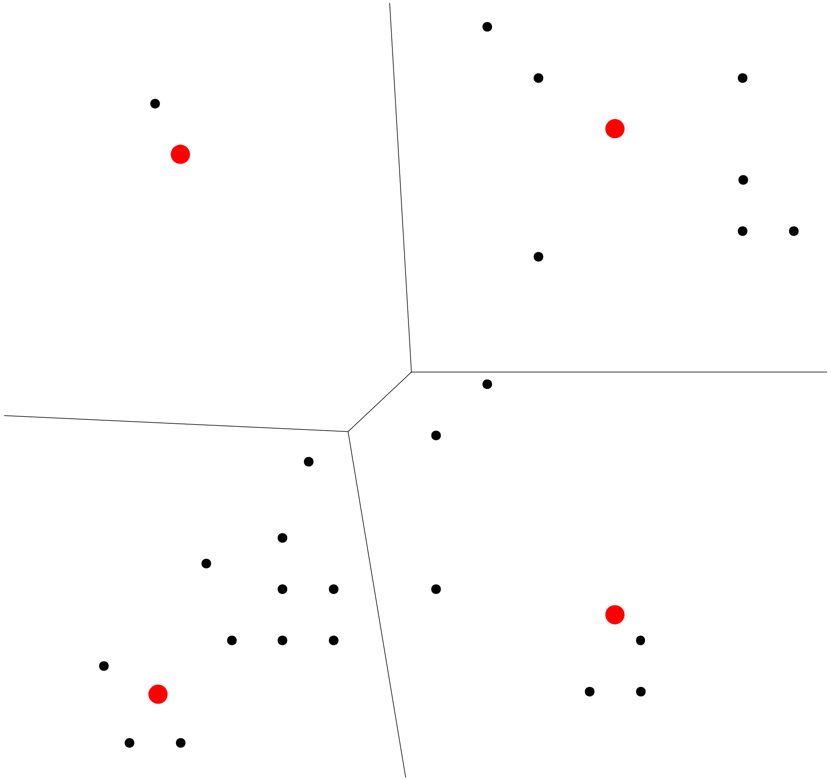
The setting

- Voters and parties are 2D points



The setting

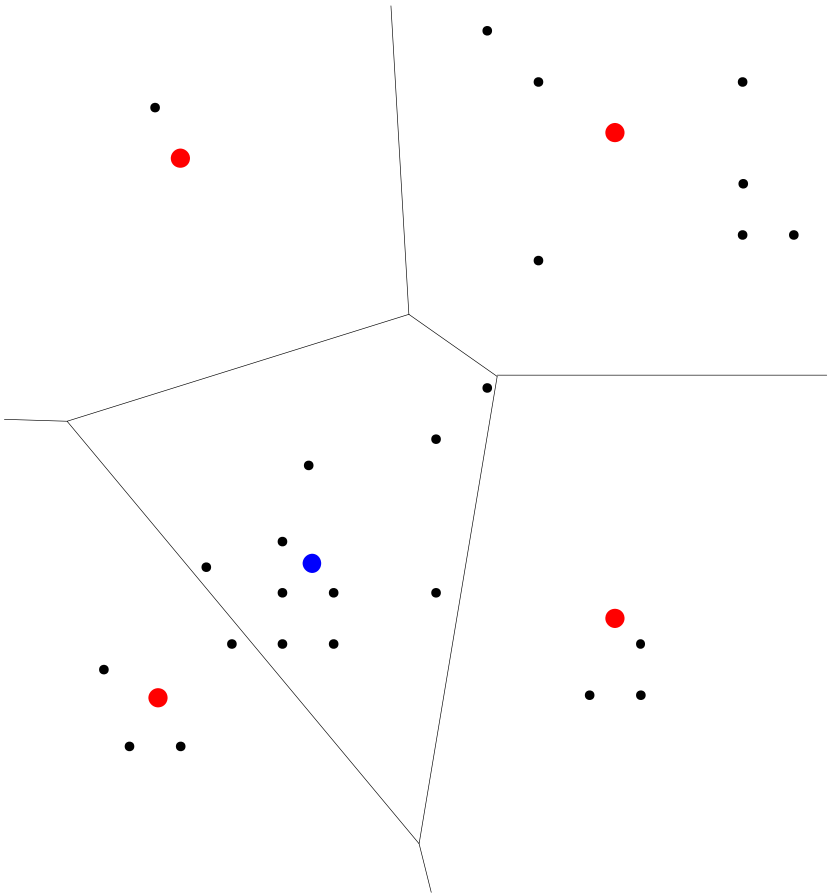
- Voters and parties are 2D points
- Voters vote for the closest party



The setting

- Voters and parties are 2D points
- Voters vote for the closest party

- How many voters can Player 1 win with a good strategy?



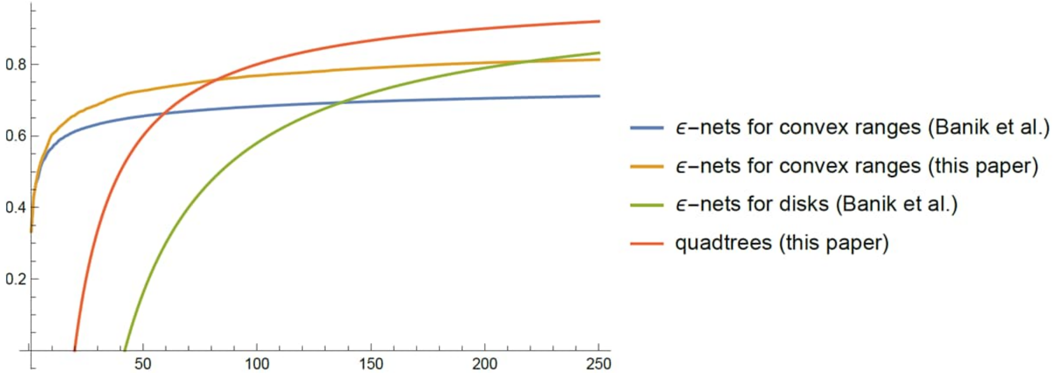
Previous work

Banik et al. (2016)

- When Player 1 optimally places k points, then Player 2 can place 1 point to win between $\frac{1}{2k}n$ and $\frac{42}{k}n$ voters
- For $k \leq 136$, Player 1 can place an *ε-net w.r.t. convex ranges* for the voter set to get better guarantees
  - For $k \geq 5$ this guarantees Player 1 wins more than half of the voters

New results

- New (small) ε-net construction for convex ranges
  - For $k = 4$ Player 1 can win at least half of the voters*
- Upper bound on voters won by Player 2 improved to $\frac{39}{k}n$ with new quadtree-based technique
- Further improved to $\frac{20\frac{5}{8}}{k}n + 6$ by combining quadtrees and ε-nets



*Assuming n is even and general position

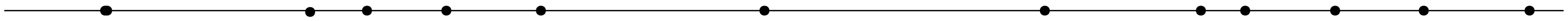
(Weak) ε-nets

- For a point set V, an *ε-net* w.r.t. some *range space* ensures that any range containing more than ε of the points of V must intersect the ε-net

(Weak) ε-nets

- For a point set V, an *ε-net* w.r.t. some *range space* ensures that any range containing more than ε of the points of V must intersect the ε-net

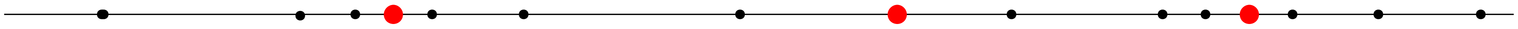
- For points on the line, the lower quartile, median and upper quartile together form a ¼-net w.r.t. intervals



(Weak) ε-nets

- For a point set V, an *ε-net* w.r.t. some *range space* ensures that any range containing more than ε of the points of V must intersect the ε-net

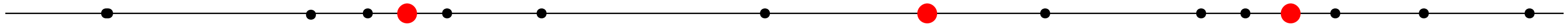
- For points on the line, the lower quartile, median and upper quartile together form a ¼-net w.r.t. intervals



(Weak) ε-nets

- For a point set V, an *ε-net* w.r.t. some *range space* ensures that any range containing more than ε of the points of V must intersect the ε-net

- For points on the line, the lower quartile, median and upper quartile together form a $\frac{1}{4}$-net w.r.t. intervals

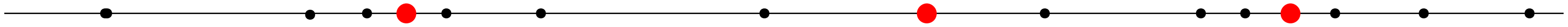


- For points in the plane, the *centerpoint* is a $\frac{2}{3}$-net w.r.t. convex sets

(Weak) ε-nets

- For a point set V, an *ε-net* w.r.t. some *range space* ensures that any range containing more than ε of the points of V must intersect the ε-net

- For points on the line, the lower quartile, median and upper quartile together form a ¼-net w.r.t. intervals



- For points in the plane, the *centerpoint* is a ⅔-net w.r.t. convex sets

- The voters won by Player 2 are in a (convex) Voronoi cell that does not intersect Player 1's points

(Weak) ε-nets

- For a point set V, an ε*-net* w.r.t. some *range space* ensures that any range containing more than ε of the points of V must intersect the ε-net

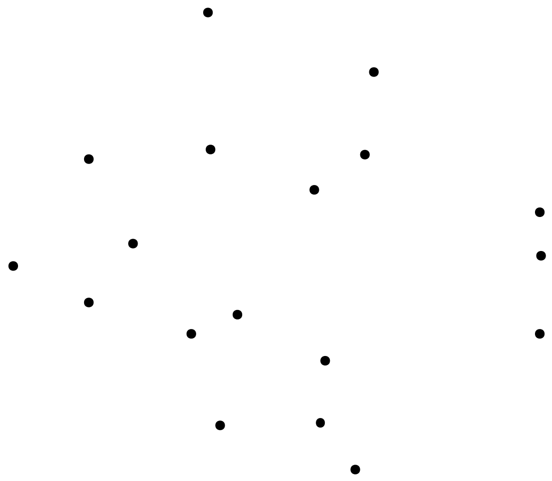
- For points on the line, the lower quartile, median and upper quartile together form a ¼-net w.r.t. intervals

- For points in the plane, the *centerpoint* is a ⅔-net w.r.t. convex sets

- The voters won by Player 2 are in a (convex) Voronoi cell that does not intersect Player 1's points
  - Thus, Player 1 can use an ε-net to ensure Player 2 wins at most εn voters

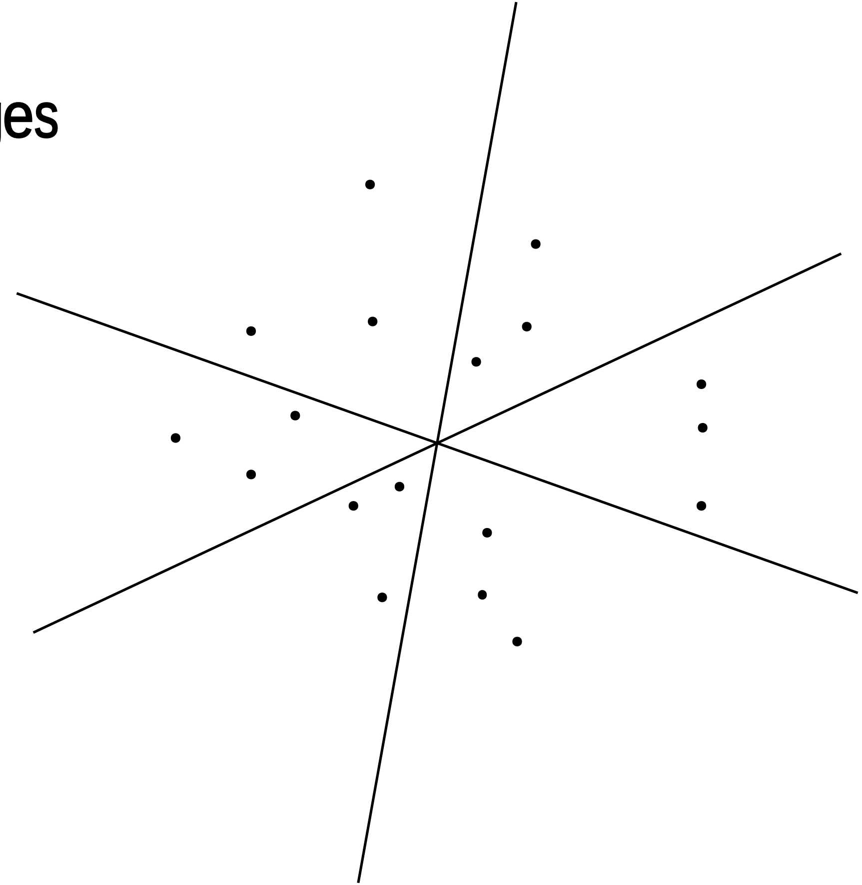
Making a ½-net w.r.t. convex ranges

- Set V of n points in general position



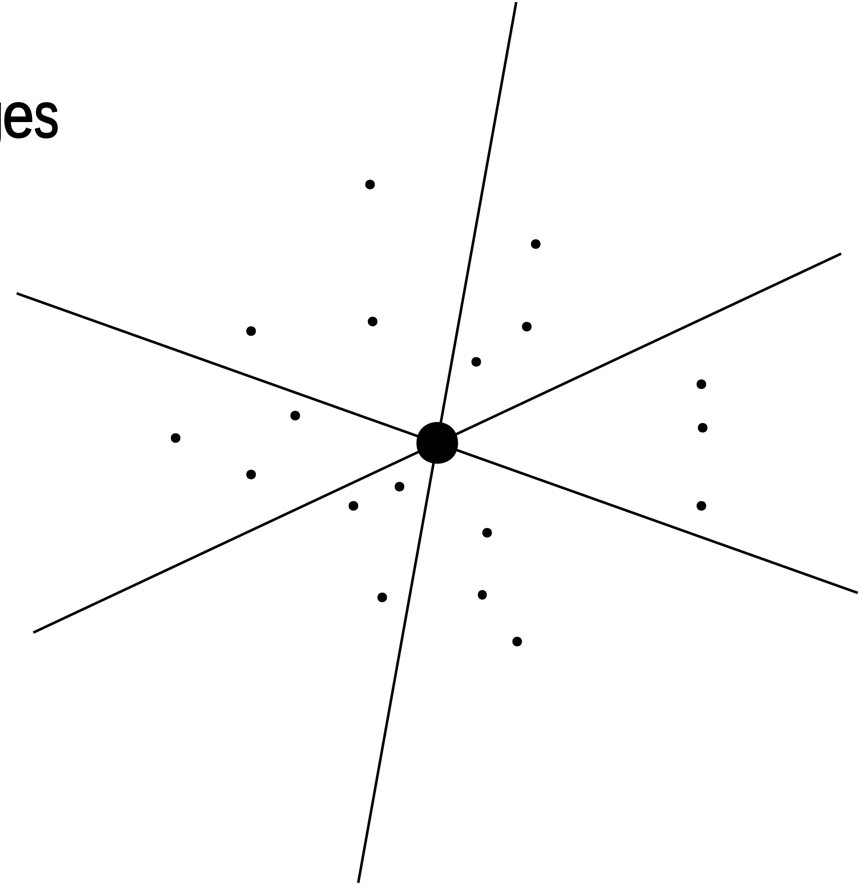
Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V



Making a ½-net w.r.t. convex ranges

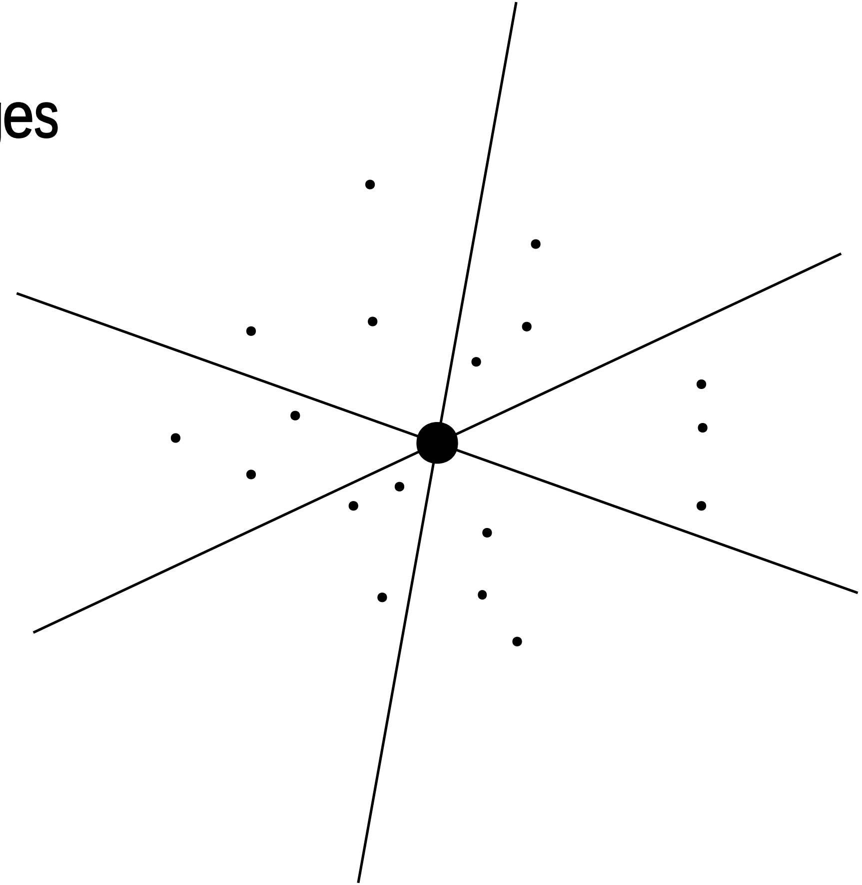
- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V
- Place a point at the intersection



Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V

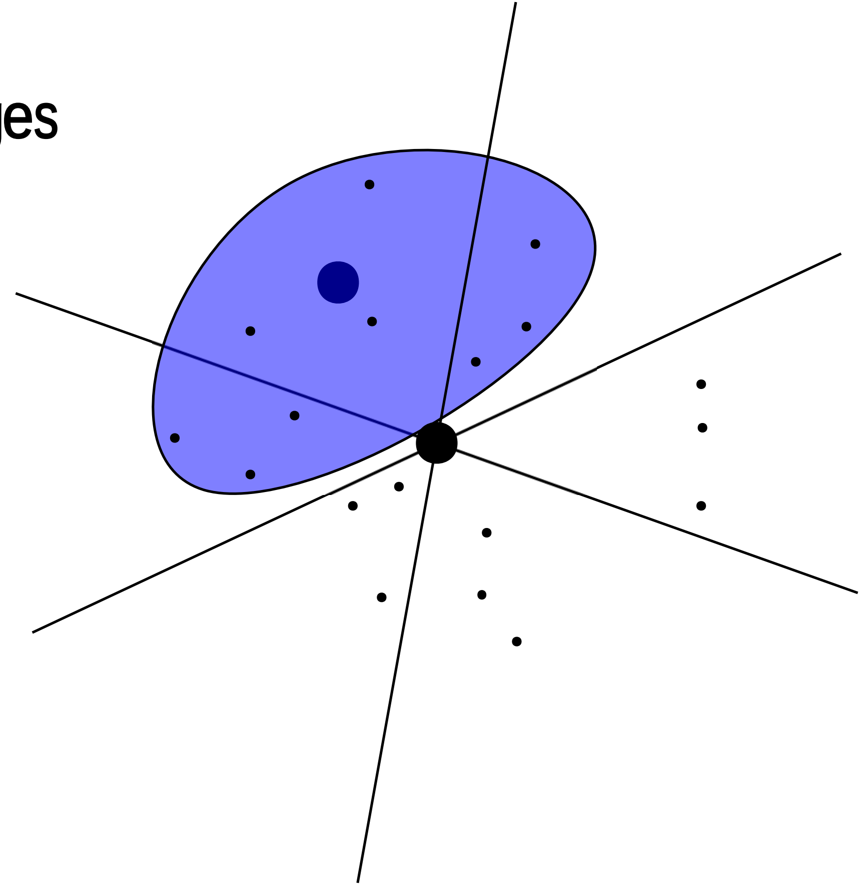
- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges



Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V

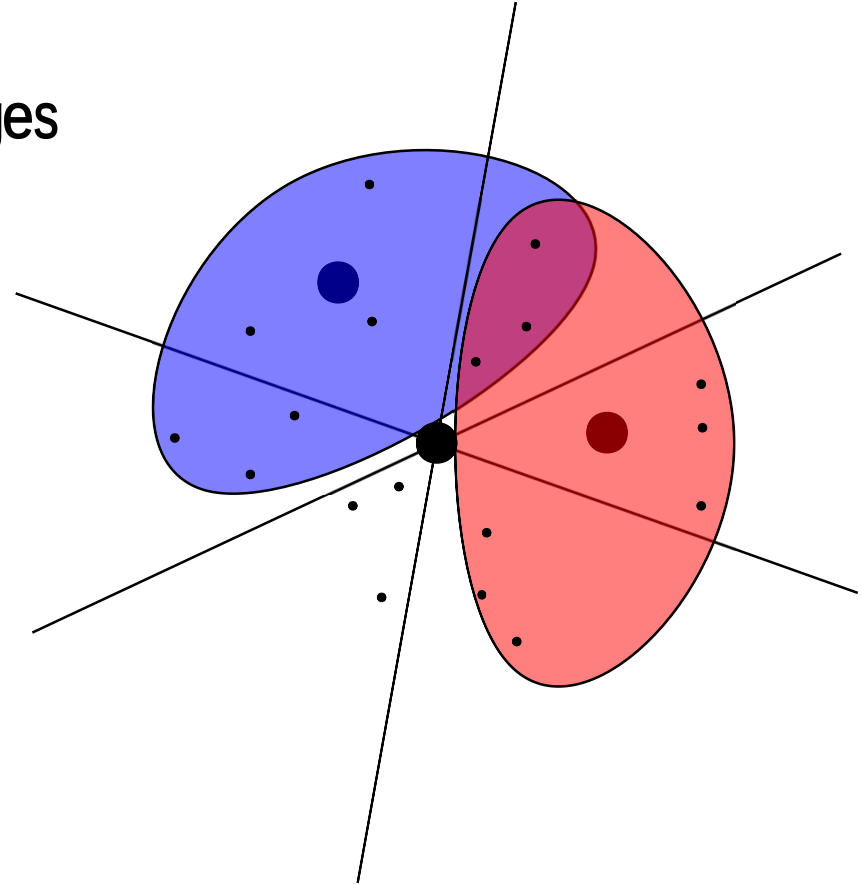
- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges



Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V

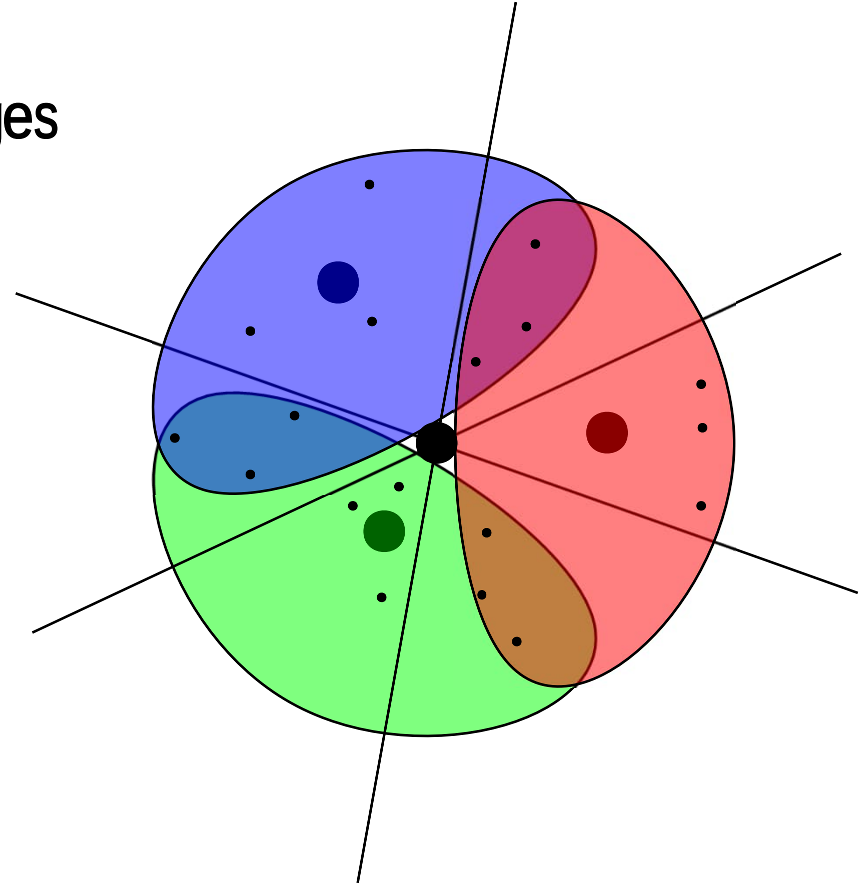
- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges



Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V

- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges

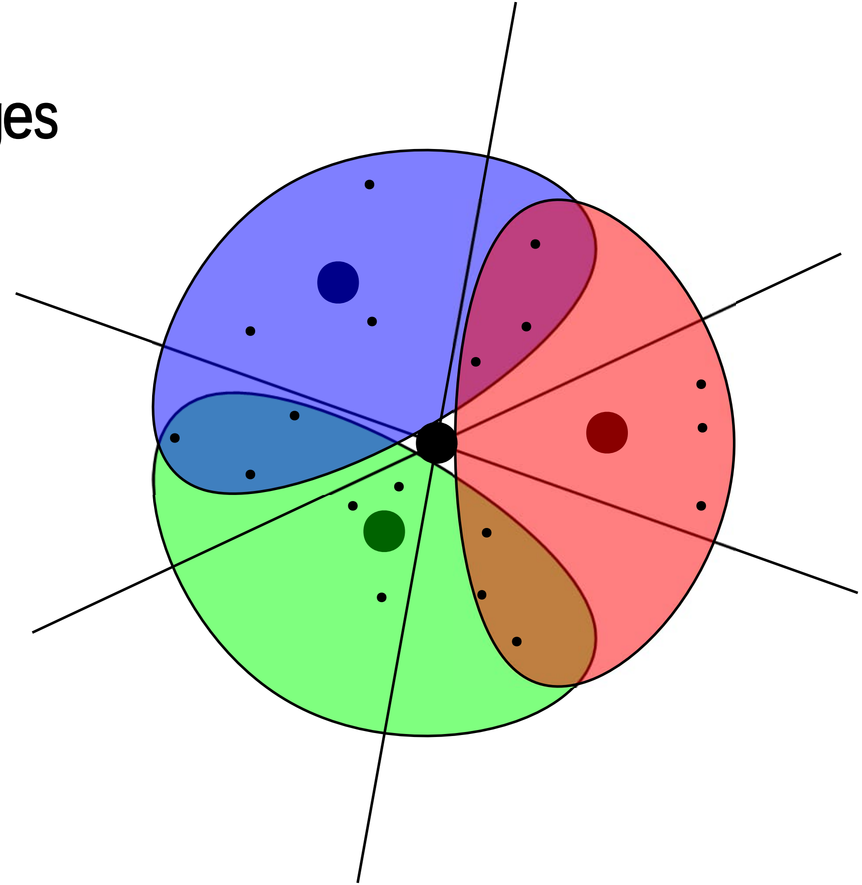


Making a ½-net w.r.t. convex ranges

- Set *V* of *n* points in general position
- If *n* is divisible by 6, three concurrent lines can equipartition *V*

- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges

- Any convex set not intersecting the net overlaps at most 4 adjacent wedges

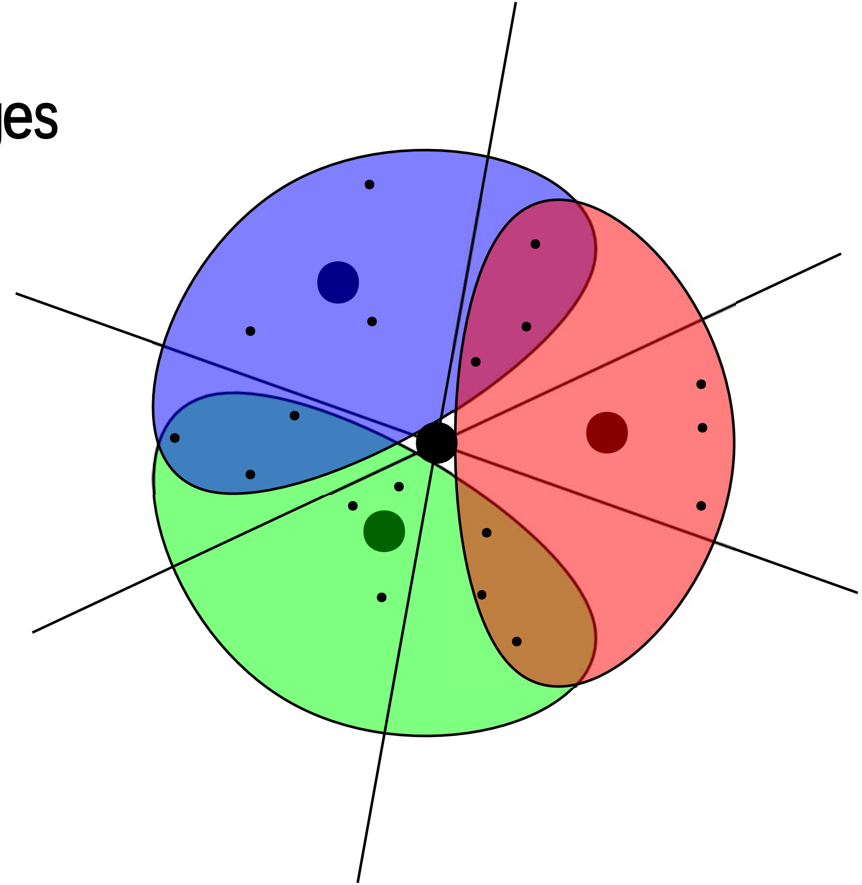


Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V

- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges

- Any convex set not intersecting the net overlaps at most 4 adjacent wedges
- Three of these will share a centerpoint

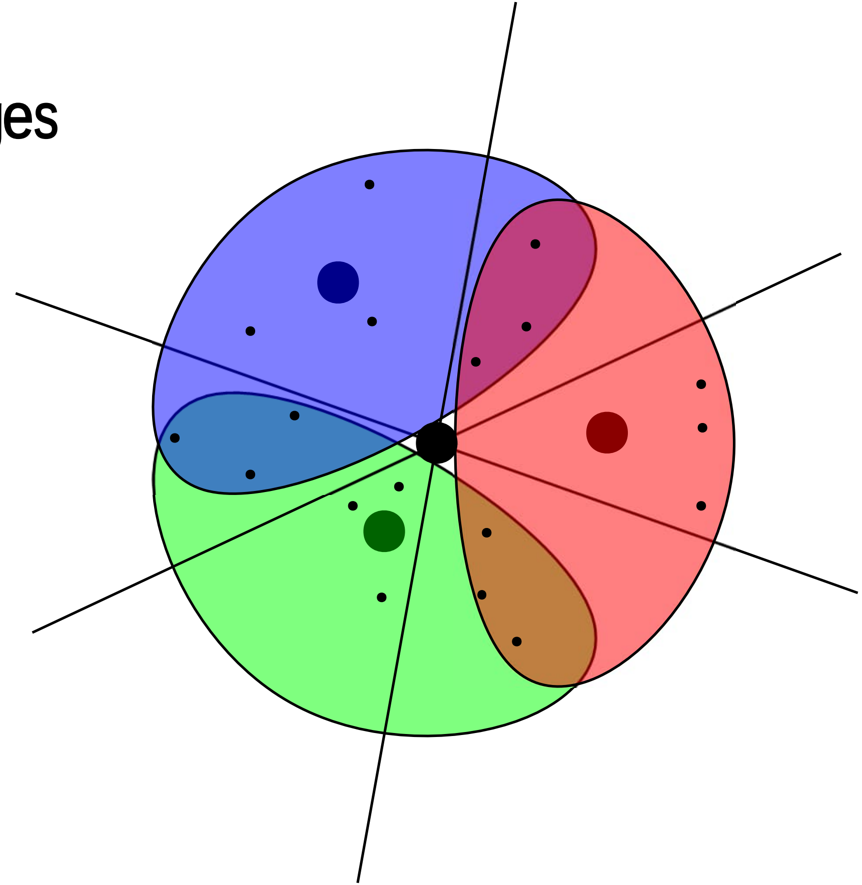


Making a ½-net w.r.t. convex ranges

- Set V of n points in general position
- If n is divisible by 6, three concurrent lines can equipartition V

- Place a point at the intersection
- Place centerpoints for the 3 combinations of 3 wedges

- Any convex set not intersecting the net overlaps at most 4 adjacent wedges
- Three of these will share a centerpoint
- It overlaps at most $\frac{n}{6} + \frac{2}{3} \cdot 3 \cdot \frac{n}{6} = \frac{n}{2}$ points

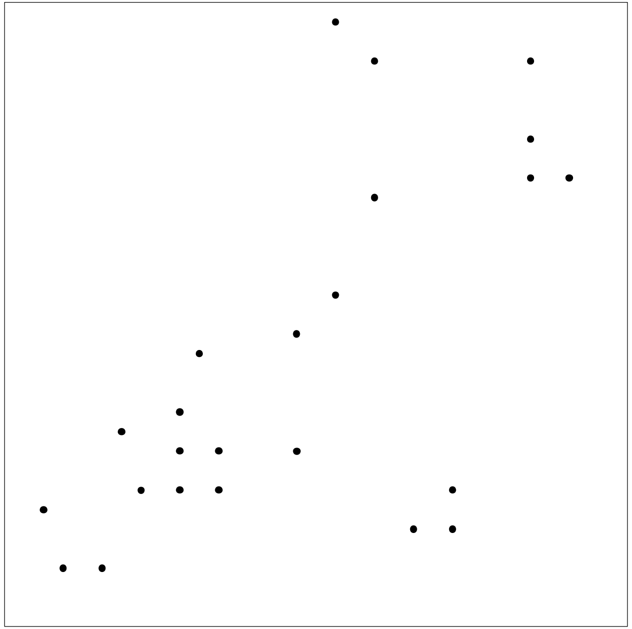


Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter

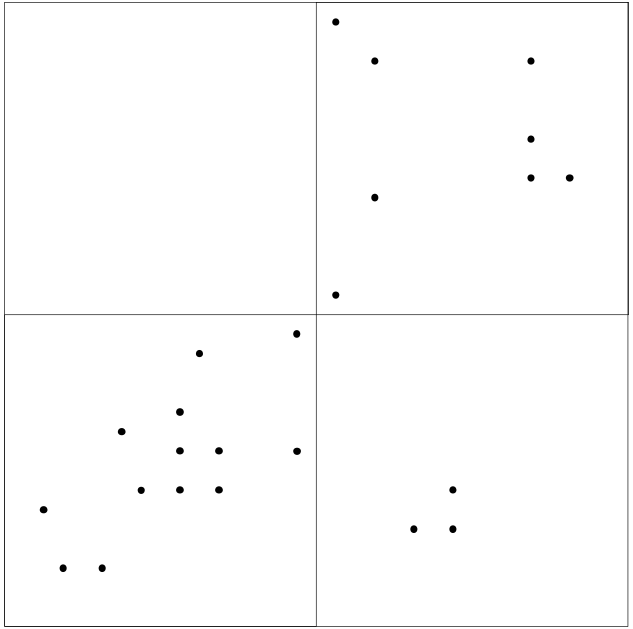
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



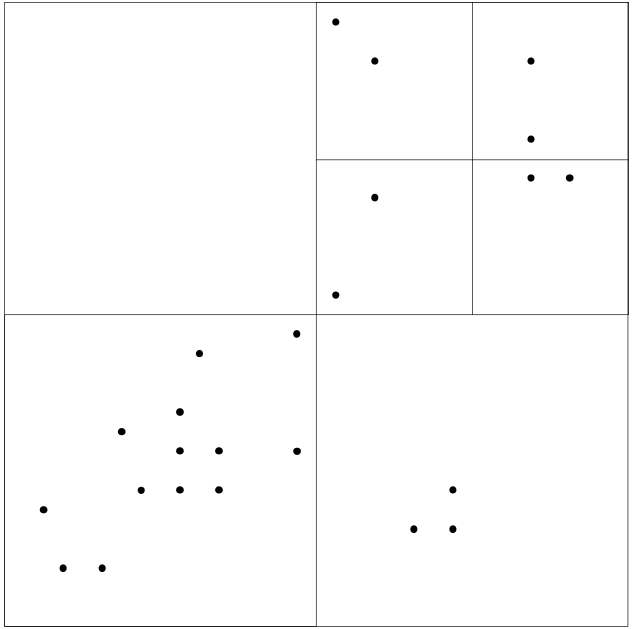
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



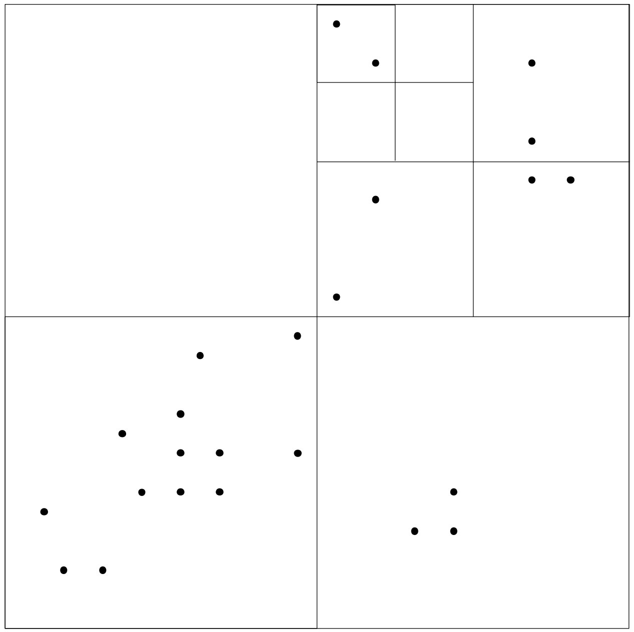
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



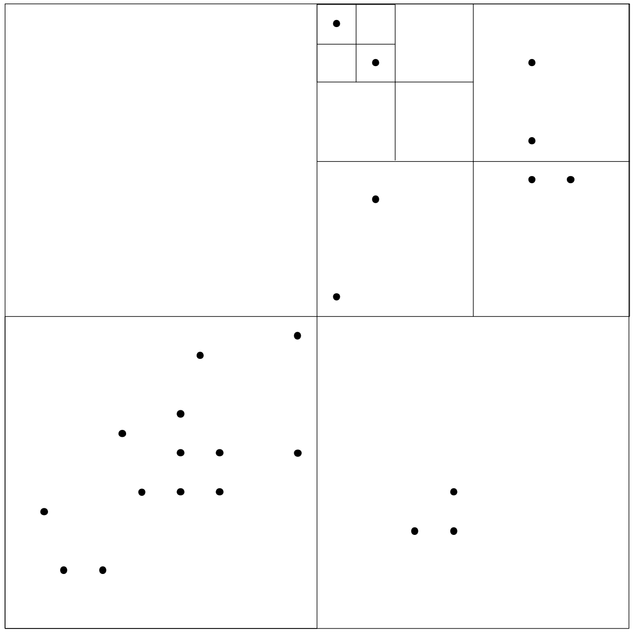
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



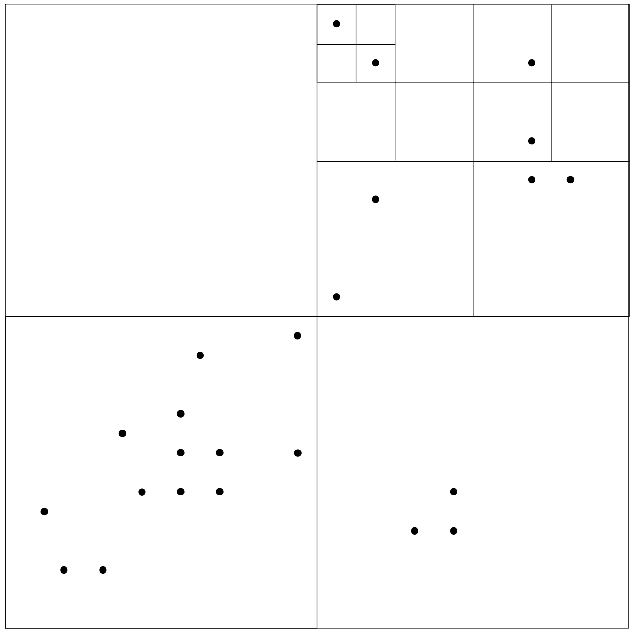
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



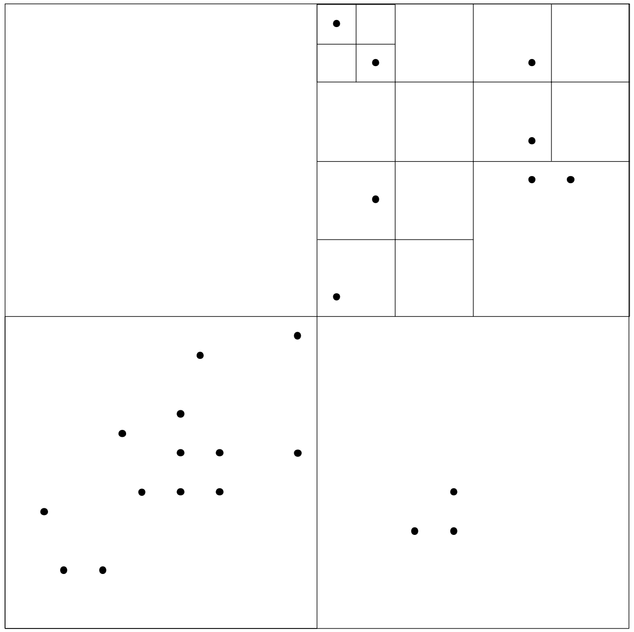
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



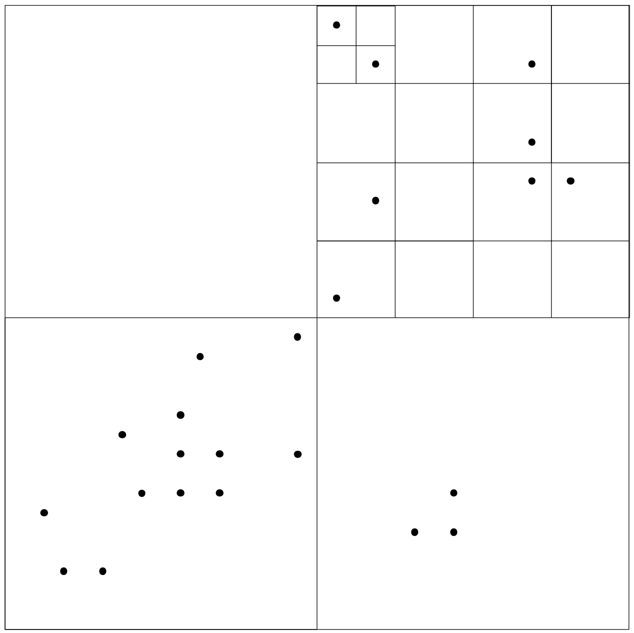
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



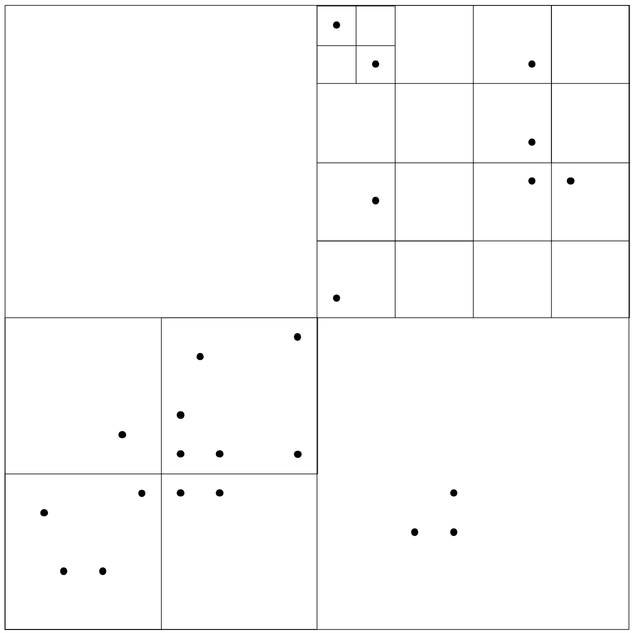
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



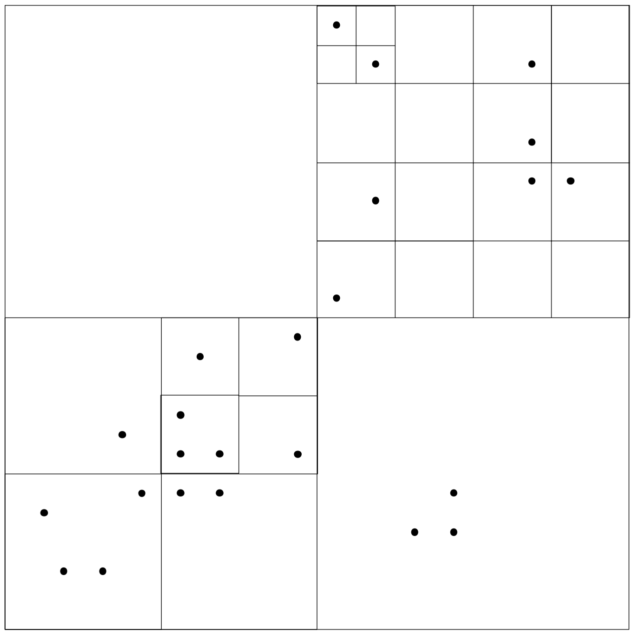
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



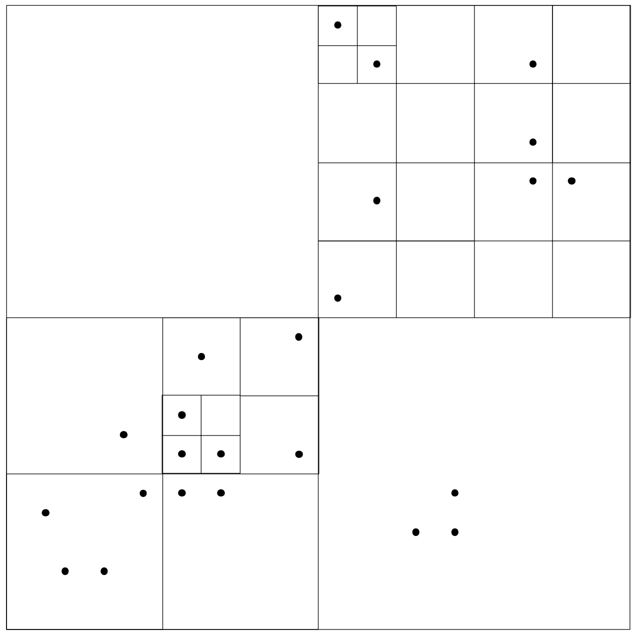
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



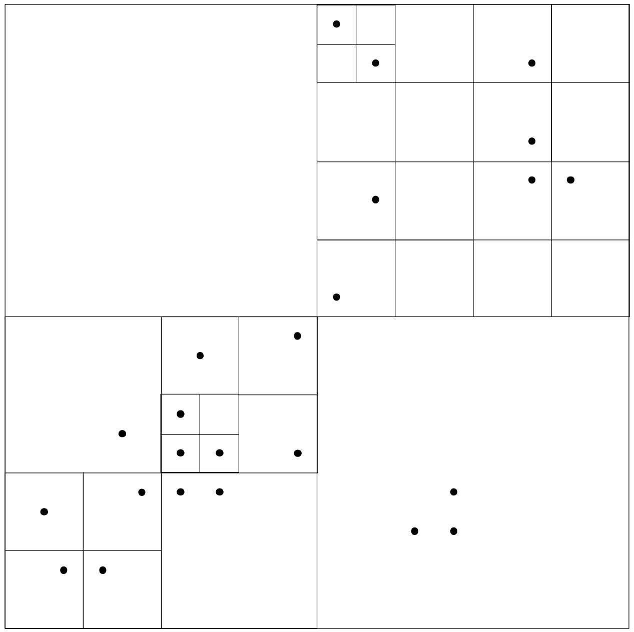
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



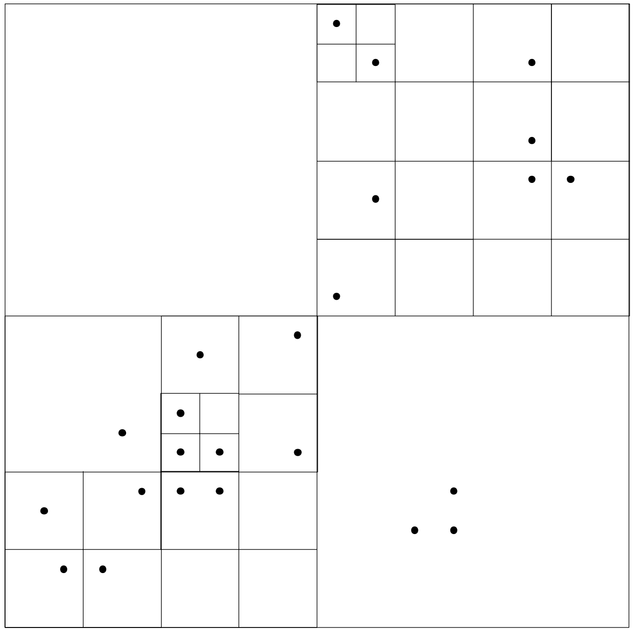
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



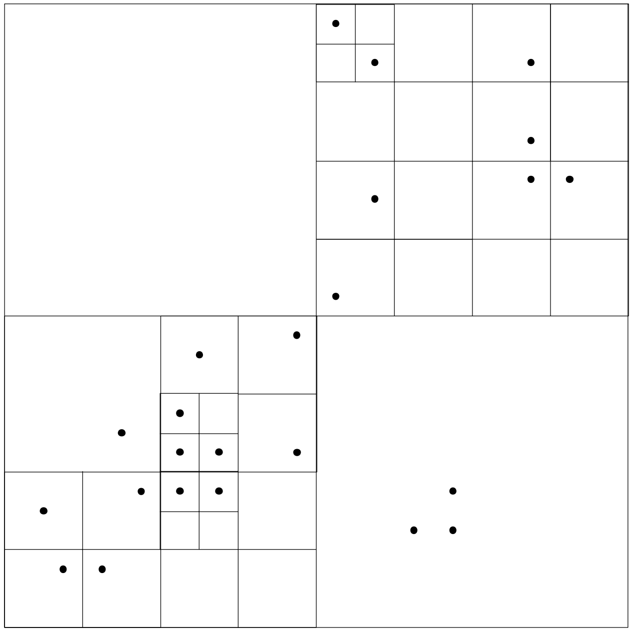
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



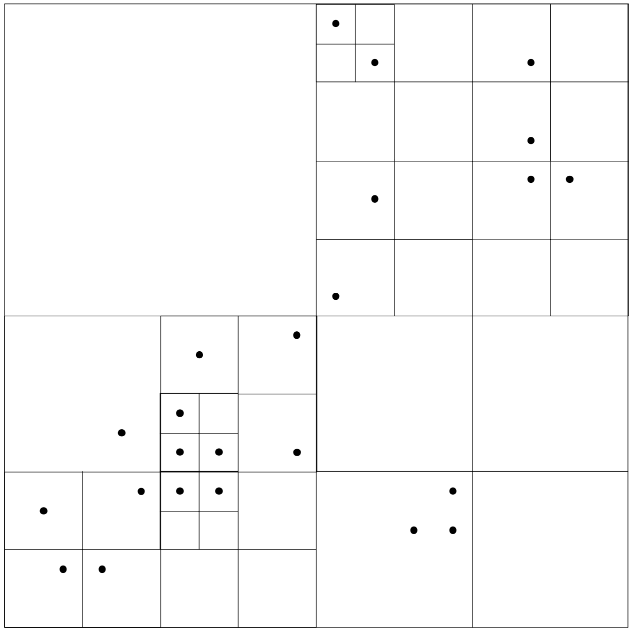
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



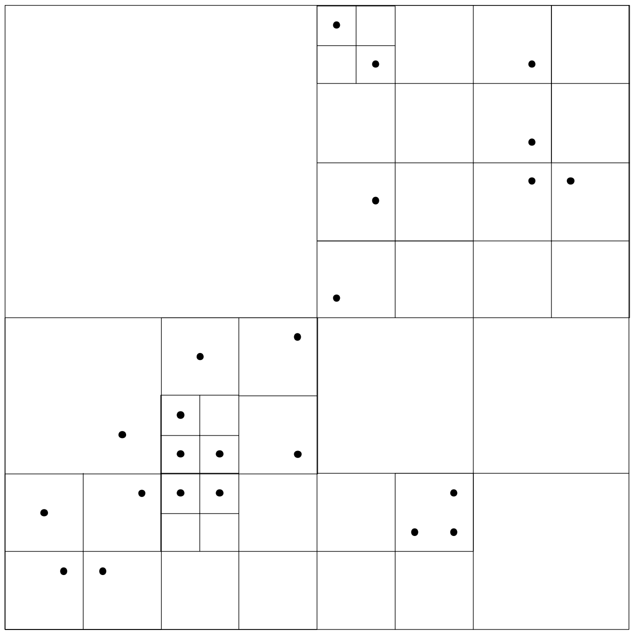
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



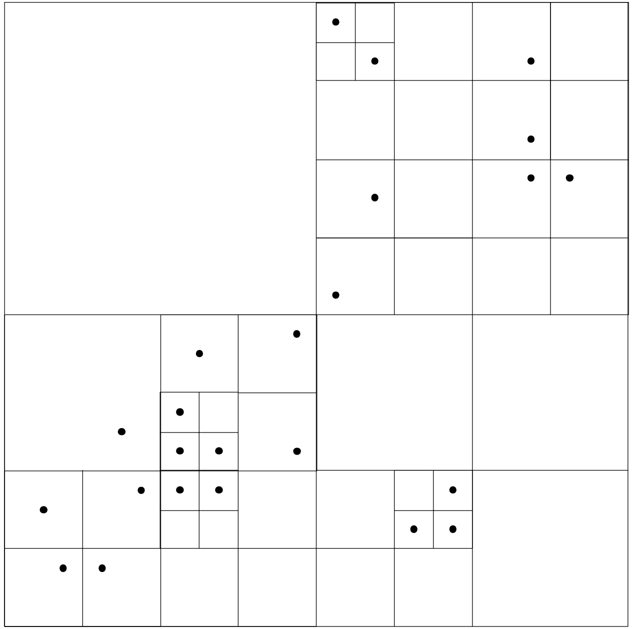
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter



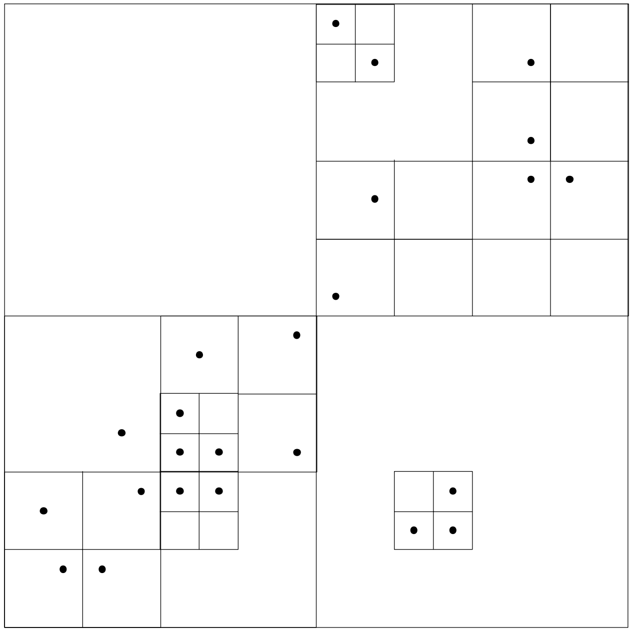
Quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter

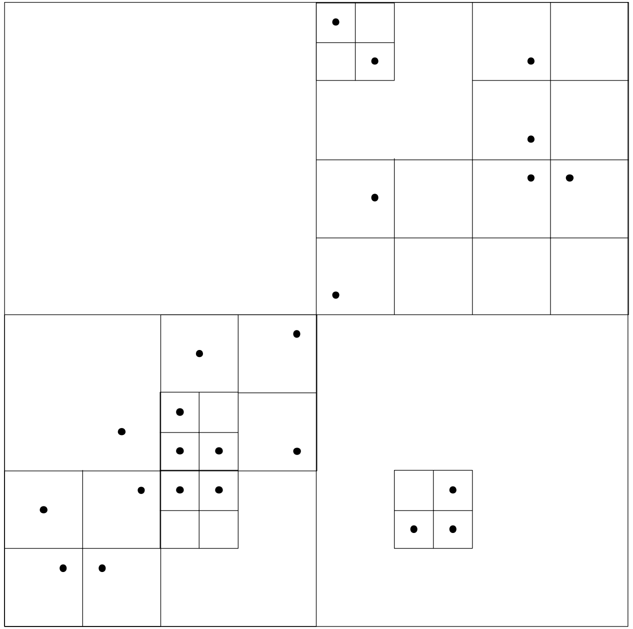


Compressed quadtree

- Has a finer grid only where there are many voters
- Subdivide squares until each contains at most one voter

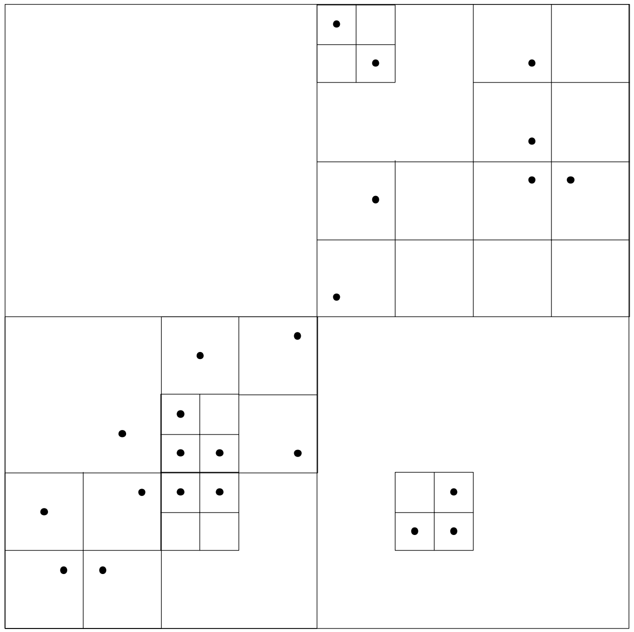


Quadtree-based technique



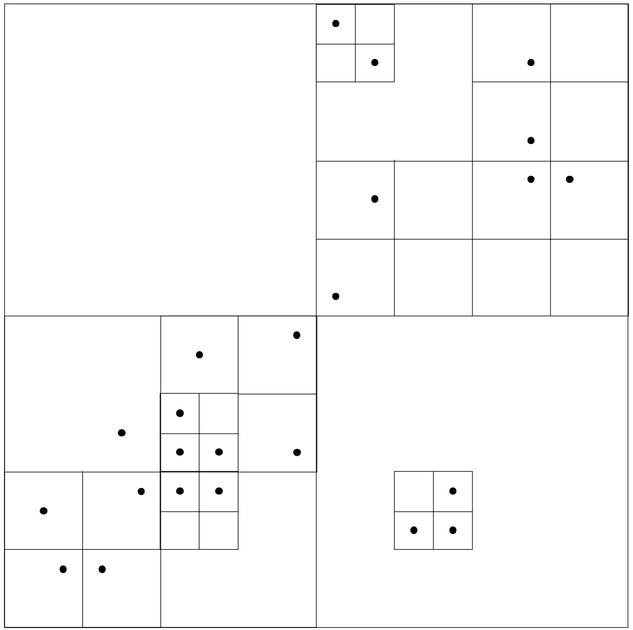
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters



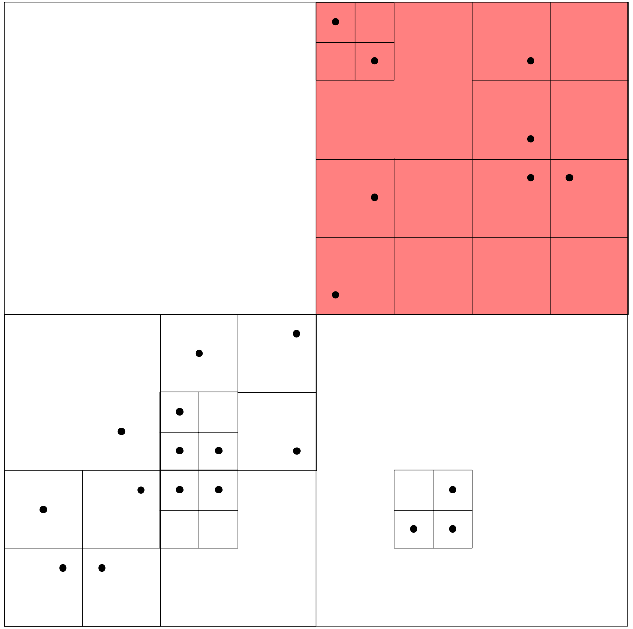
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters



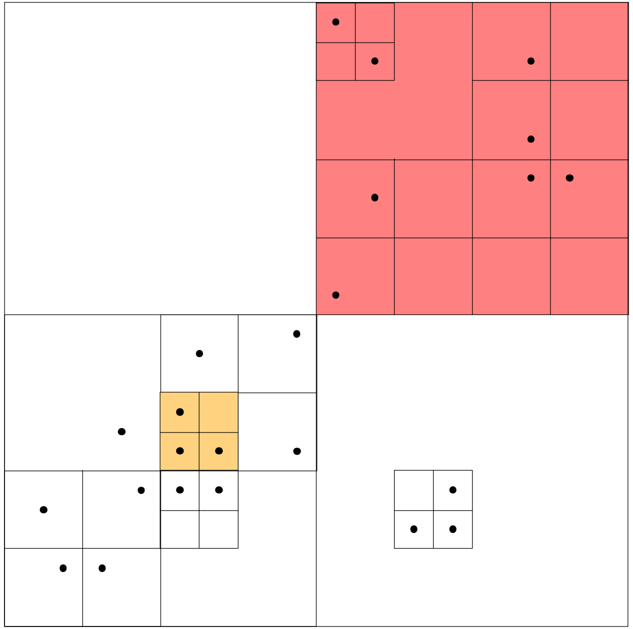
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters



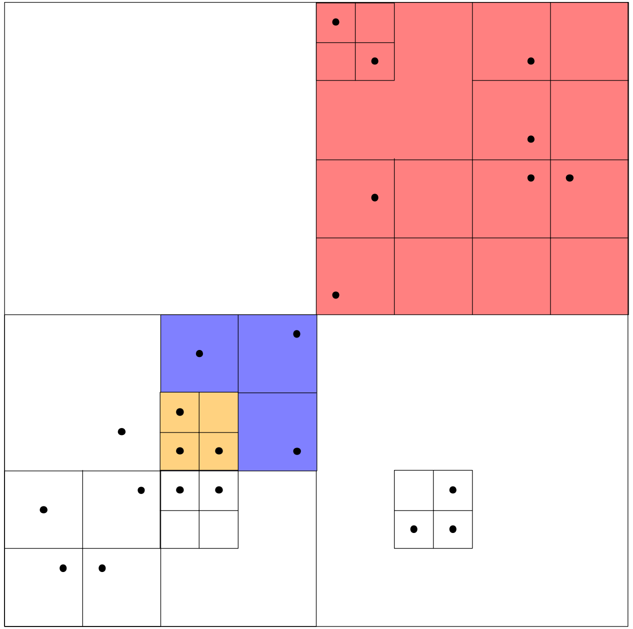
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters



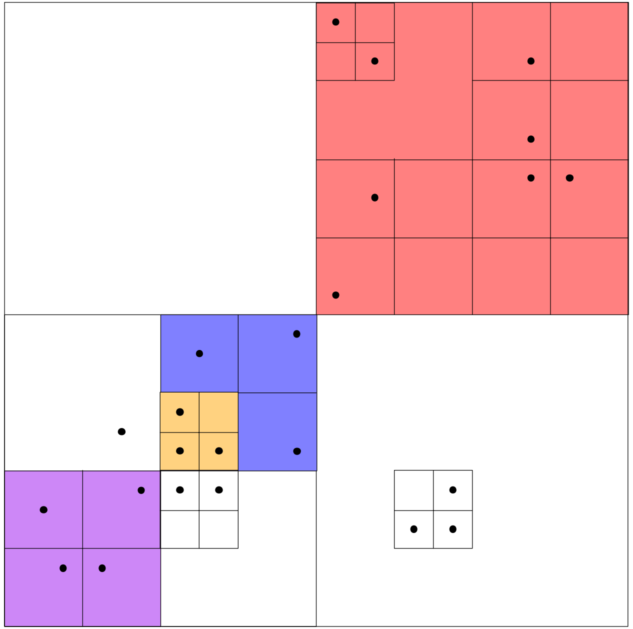
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters



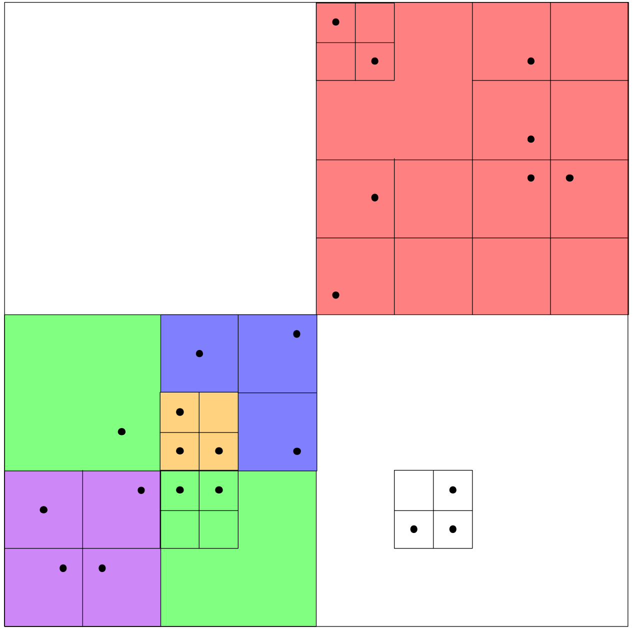
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters



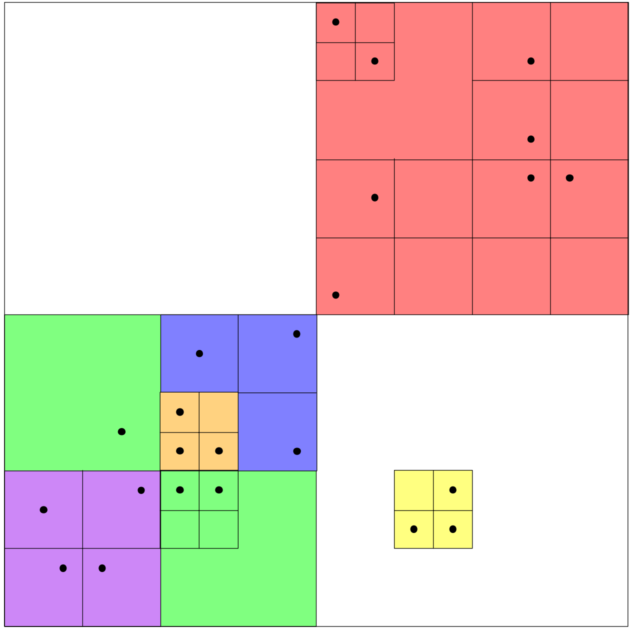
Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters

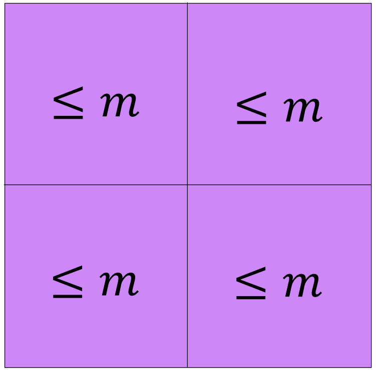


Quadtree-based technique

- Take the (compressed) quadtree of the voter set
- Set a parameter m (here $m = 2$)
- Starting from the leaves, select a cell if it contains more than m not yet covered voters
- Each region is a selected cell without its selected descendants, and covers between $m + 1$ and $4m$ voters

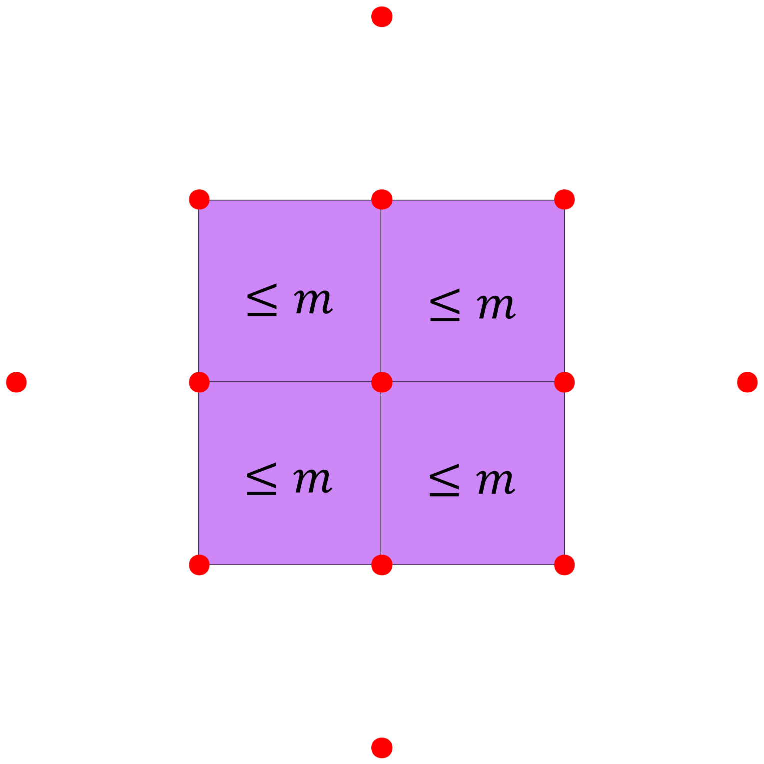


Placing points



Placing points

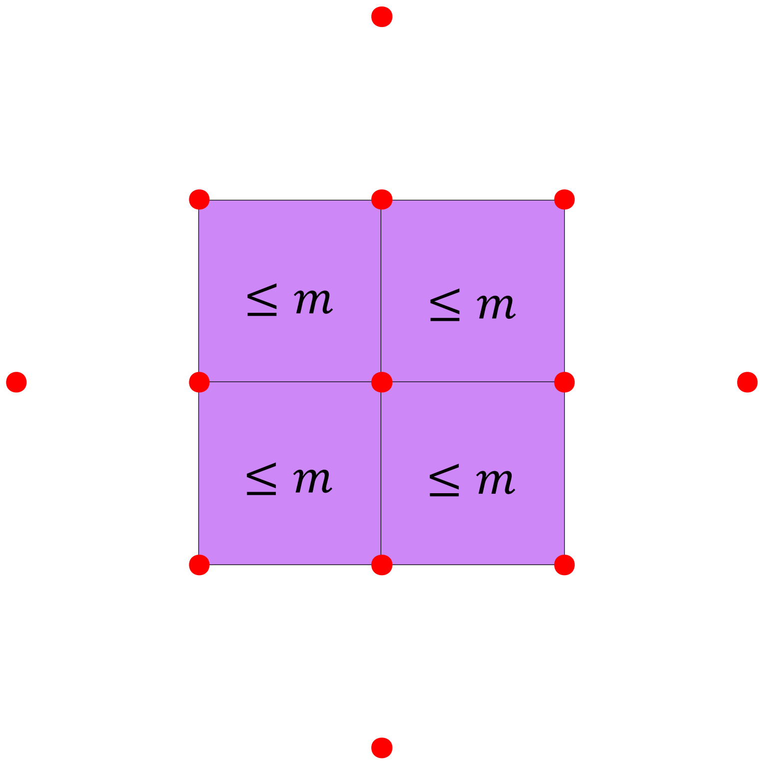
- Placing 13 points as shown ensures Player 2 wins from at most 3 'child' cells



Placing points

- Placing 13 points as shown ensures Player 2 wins from at most 3 ‘child’ cells

- $3m < 3n/\frac{k}{13} = \frac{39}{k}n$



Conclusion

- We can use different ε-nets to place proposals
- Using quadtrees, Player 1 can always place *k* proposals such that Player 2 can win at most $\frac{39}{k}n$ voters
- By combining the two, we can make this $\frac{20\frac{5}{8}}{k}n + 6$
- Can we prove tight bounds for some $k > 1$?