The Marco Polo Problem: A Combinatorial Approach to Geometric Localization

Ofek Gila[1], Michael T. Goodrich[1], Zahra Hadizadeh[2], Daniel S. Hirschberg[1], and Shayan Taherijam[1]

[1]University of California, Irvine

[2]University of Rochester

CCCG, 2025

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin

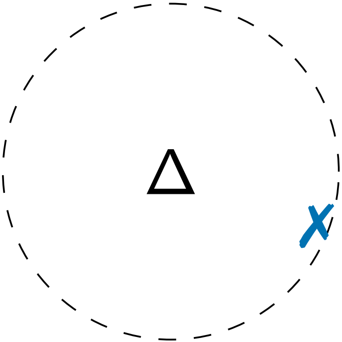


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$

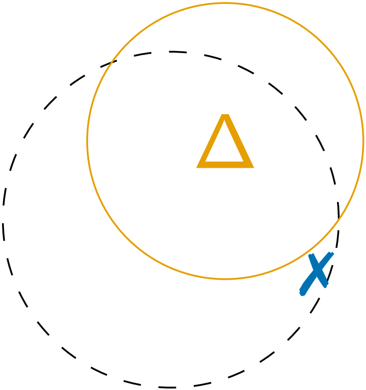


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗

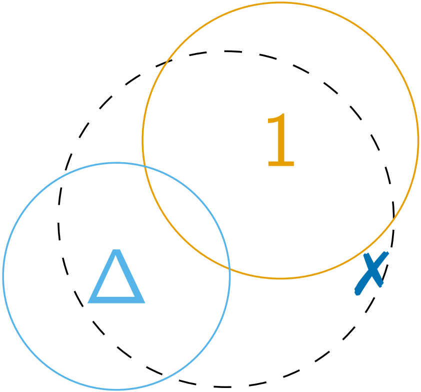


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗.

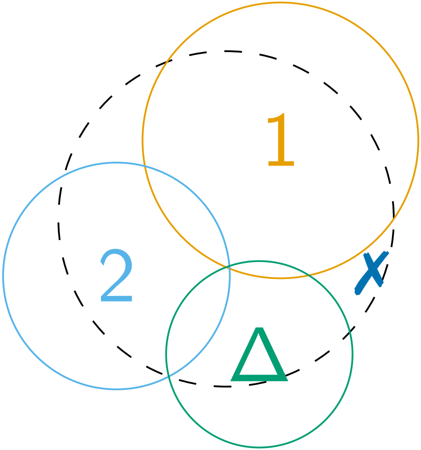


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗..

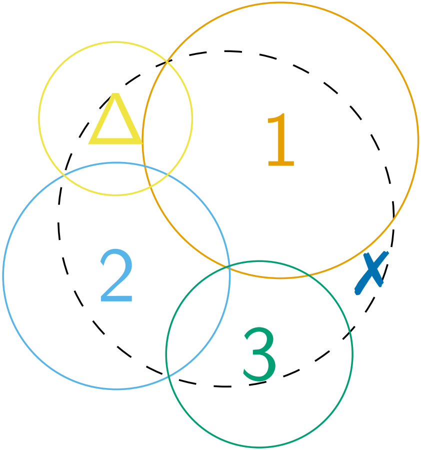


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1

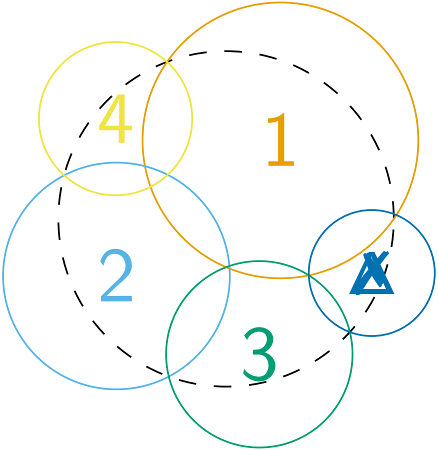


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!

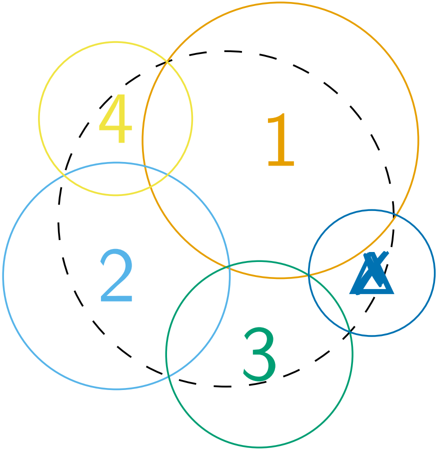


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - $\#$ of POIs present (k)

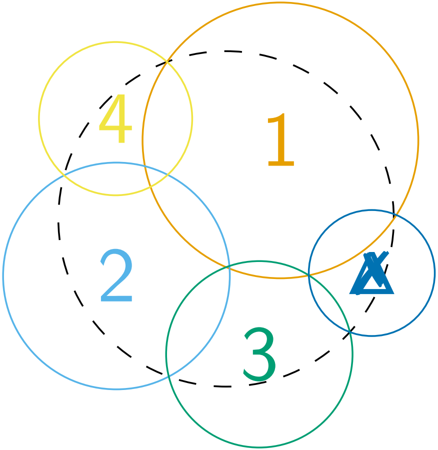


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - $\#$ of POIs present (k)
  - find all POIs

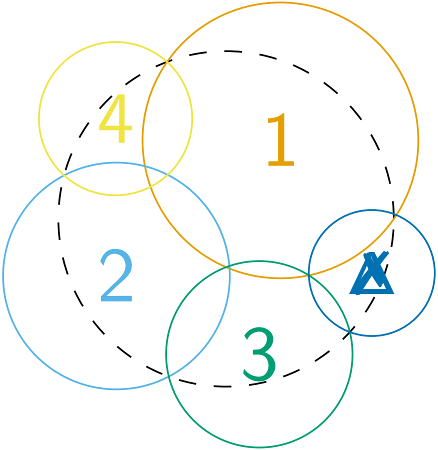


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)

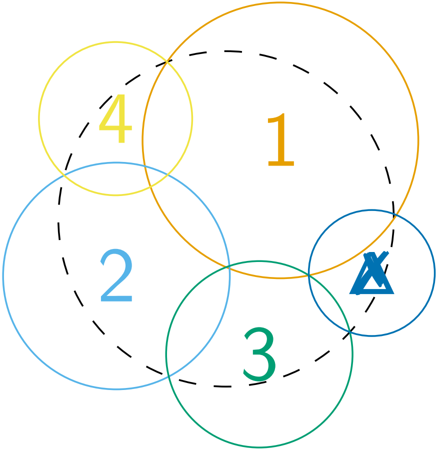


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)

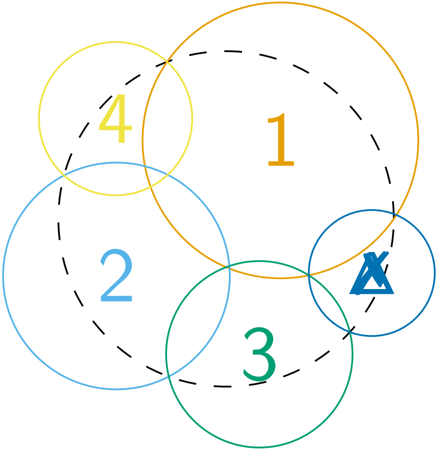


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)

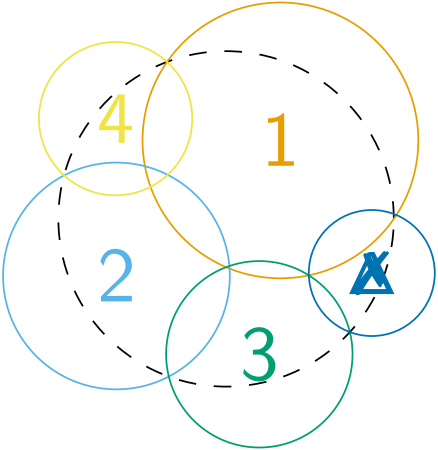


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any

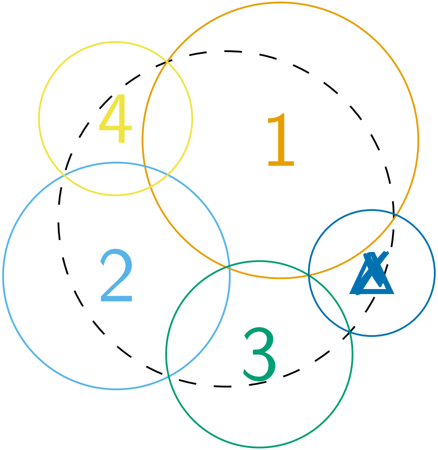


Figure 1: A search area.

The Marco Polo Problem I

- Point of Interest (POI) ✗
- ✗ within distance n from origin
- Probes with radius d, $p(x, y, d)$
- Probe until 'finding' ✗... ✓
- 'finding': distance $\Delta \leftrightarrow$ ✗ ≤ 1
- Δ must know this!
- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...

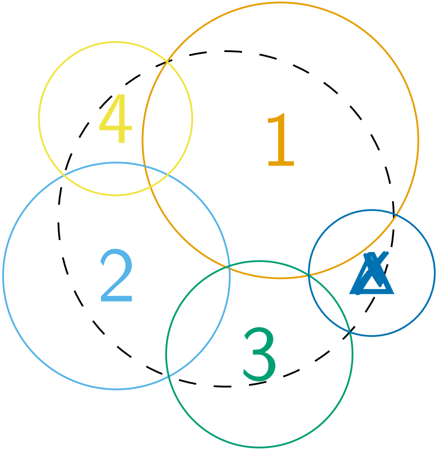


Figure 1: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...
- Effectiveness metrics:

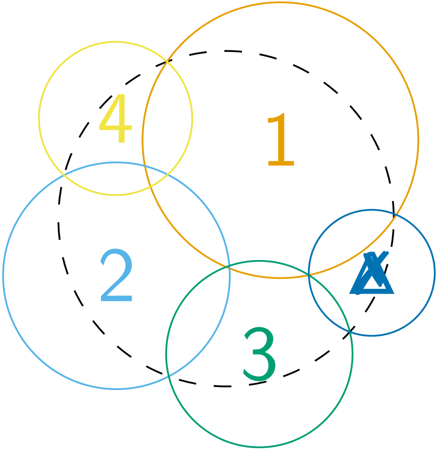


Figure 2: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...
- Effectiveness metrics:
  - # of probes, $P(n)$

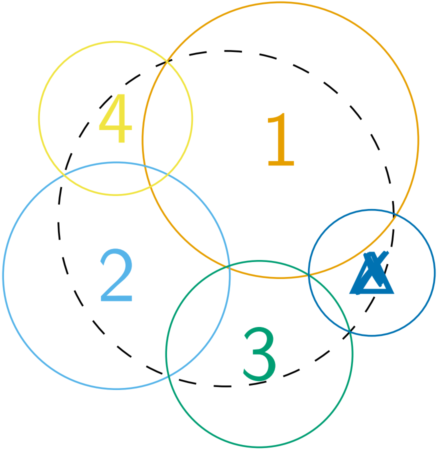


Figure 2: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...
- Effectiveness metrics:
  - # of probes, $P(n)$
  - Distance traveled by Δ, $D(n)$

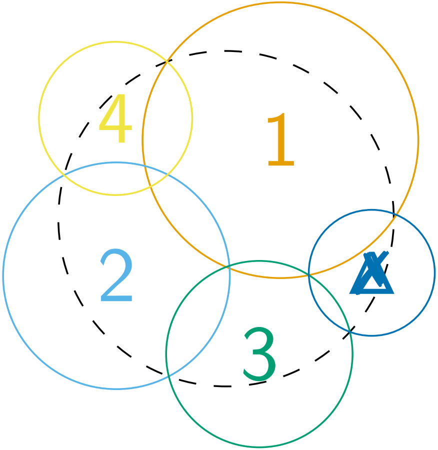


Figure 2: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...

- Effectiveness metrics:
  - # of probes, $P(n)$
  - Distance traveled by Δ, $D(n)$
  - # of POI responses, $R(n)$

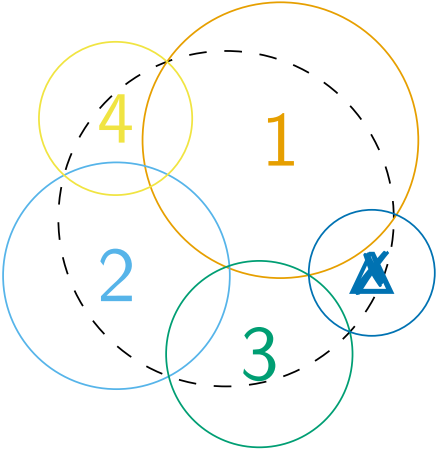


Figure 2: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...
- Effectiveness metrics:
  - # of probes, $P(n)$
  - Distance traveled by Δ, $D(n)$
  - # of POI responses, $R(n)$
  - Input sensitivity?

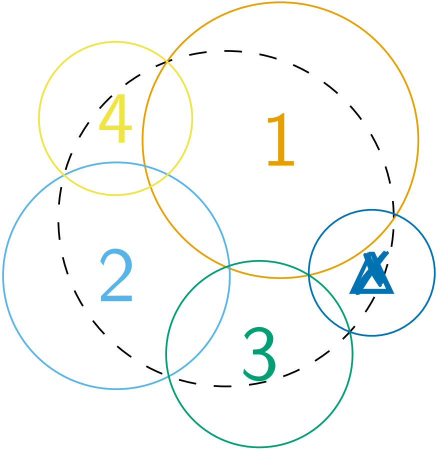


Figure 2: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...
- Effectiveness metrics:
  - # of probes, $P(n)$
  - Distance traveled by Δ, $D(n)$
  - # of POI responses, $R(n)$
  - Input sensitivity?
    - TSP tour length, $\mathtt{OPT}$

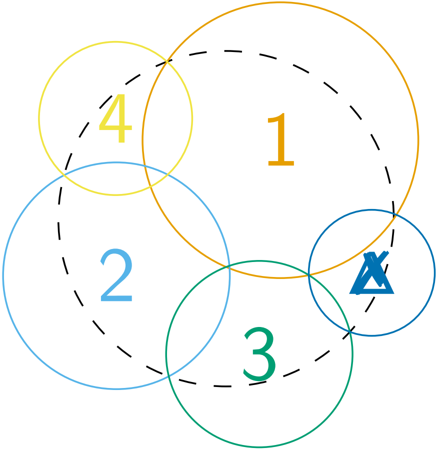


Figure 2: A search area.

The Marco Polo Problem II

- Variants:
  - # of POIs present (k)
  - find all POIs
  - Distance metrics (L_1, L_∞)
  - # of dimensions (2D, 3D, ...?)
  - Probe response (T/F, d, i)
  - Δ's memory if any
  - multiple Δ, etc...
- Effectiveness metrics:
  - # of probes, $P(n)$
  - Distance traveled by Δ, $D(n)$
  - # of POI responses, $R(n)$
  - Input sensitivity?
    - TSP tour length, `OPT`
  - Simplicity / practicality

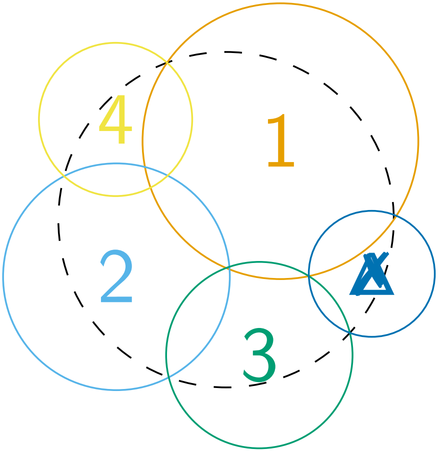


Figure 2: A search area.

- Motivation?

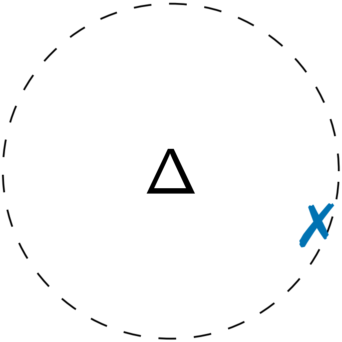


Figure 3: A search area.

Motivation

- Motivation?
- Finding gold?

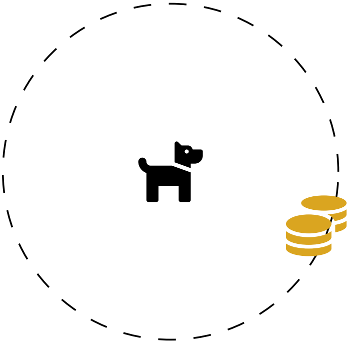


Figure 3: A search area.

Motivation

- Motivation?

- Finding gold?
- Detecting uranium?

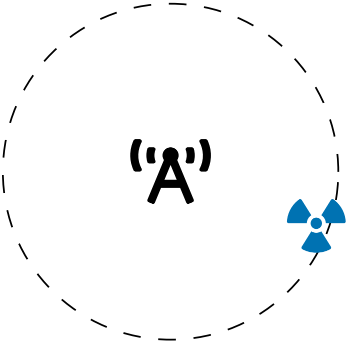


Figure 3: A search area.

Motivation

- Motivation?

- Finding gold?
- Detecting uranium?
- Finding lost hiker / kidnap victim? 

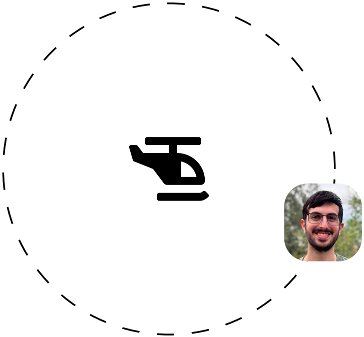


Figure 3: A search area.

Motivation

- Motivation?

- Finding gold? 
- Detecting uranium? 
- Finding lost hiker / kidnap victim? 
- Game of Marco Polo?

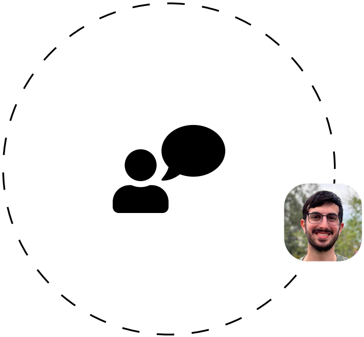


Figure 3: A search area.

Motivation

- Motivation?

- Finding gold?
- Detecting uranium?
- Finding lost hiker / kidnap victim? 
- Game of Marco Polo?

- Whatever floats your

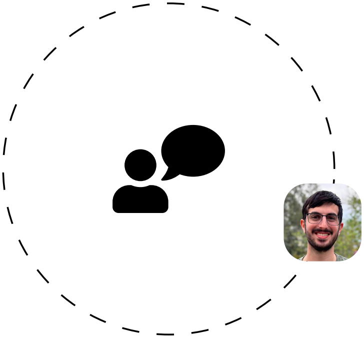


Figure 3: A search area.

Why not One?

- What if just one ✗?

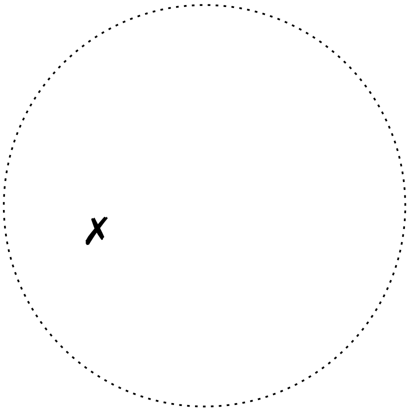


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius

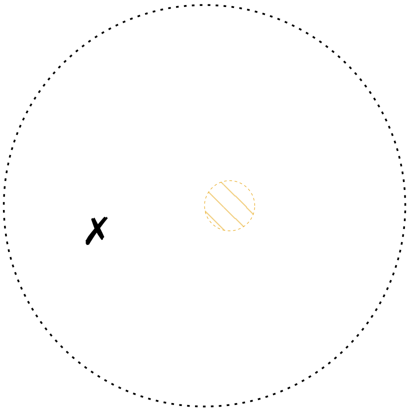


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius.

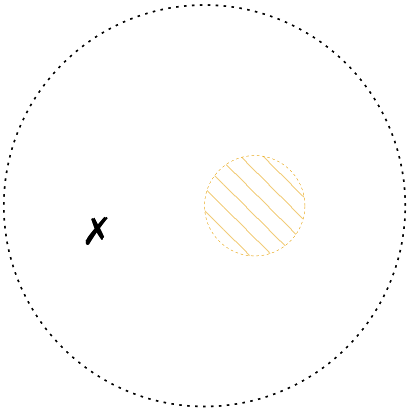


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius..

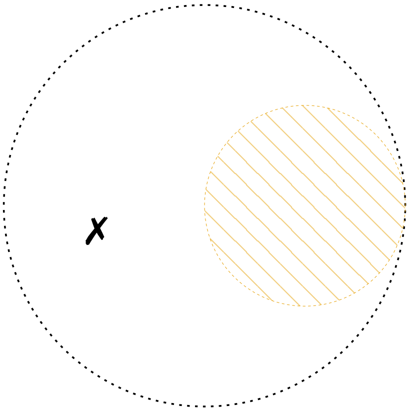


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius...

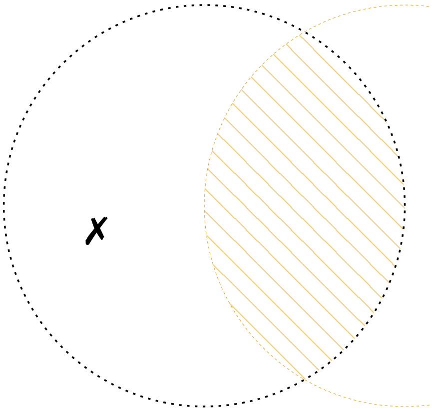


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....

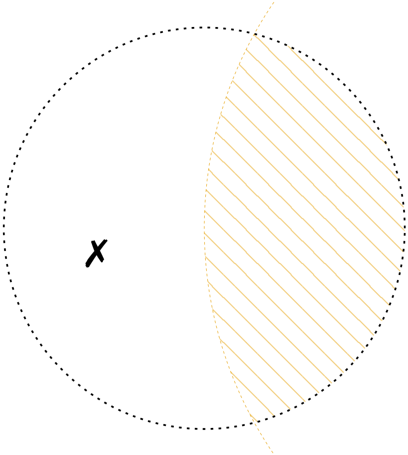


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius.....

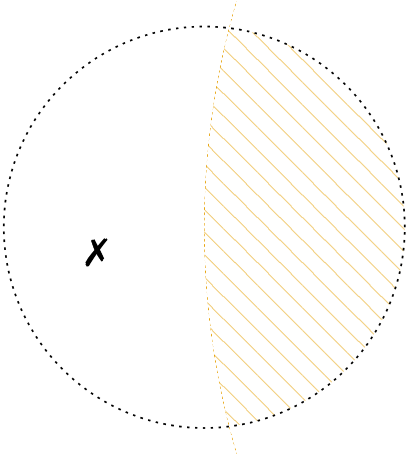


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius......

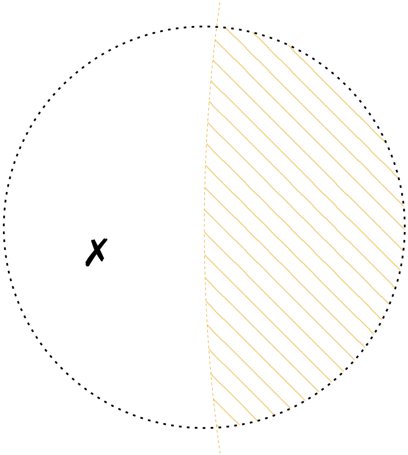


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius.......

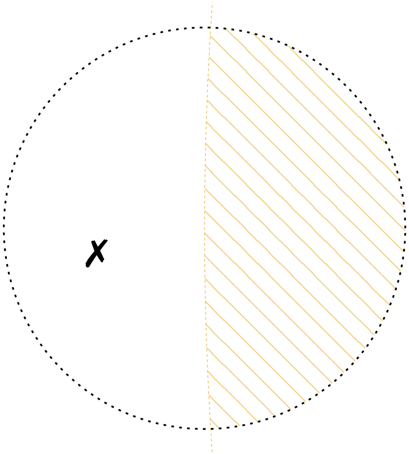


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\to \infty$

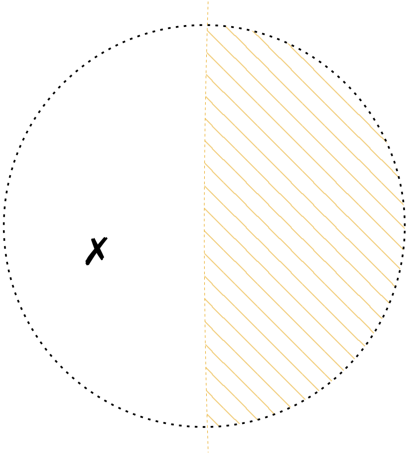


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing

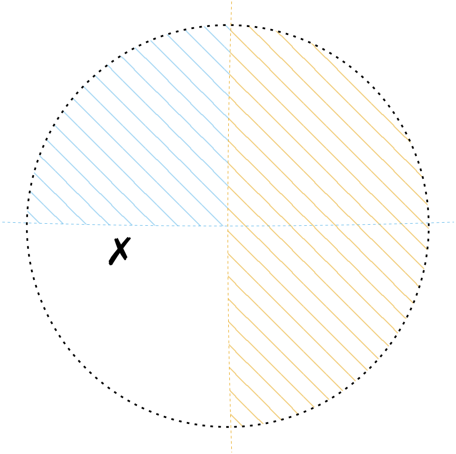


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing.

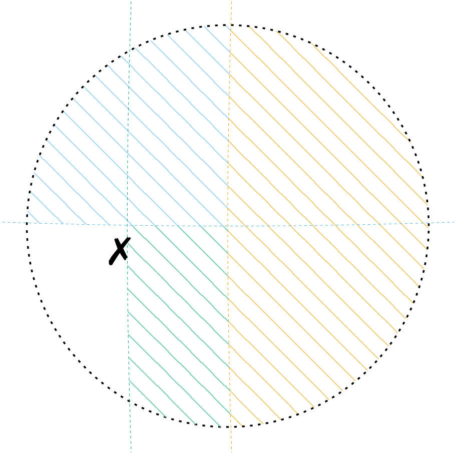


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing..

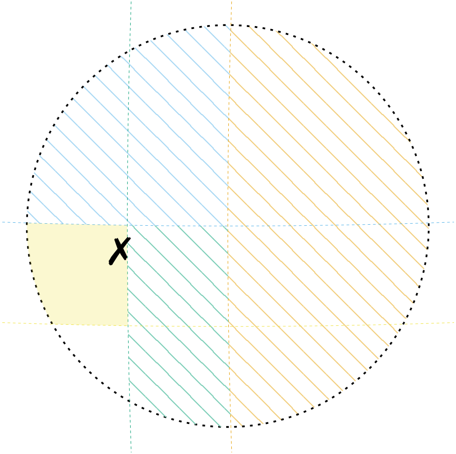


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing...

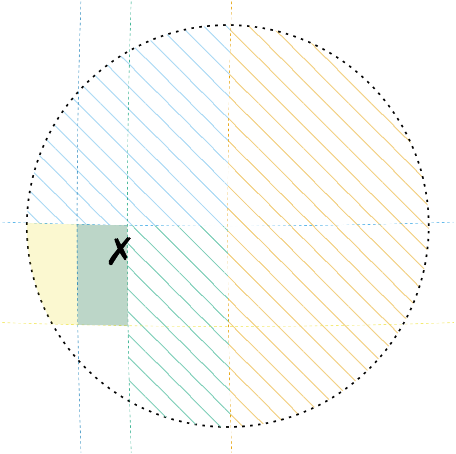


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing....

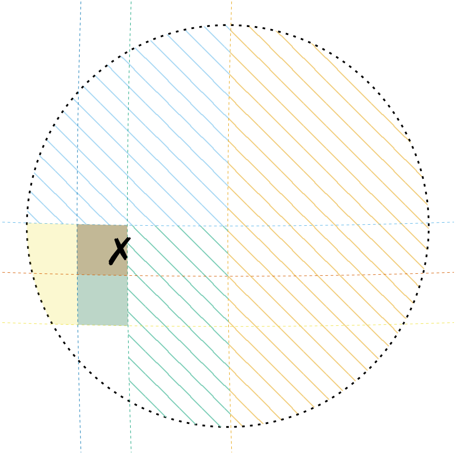


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)

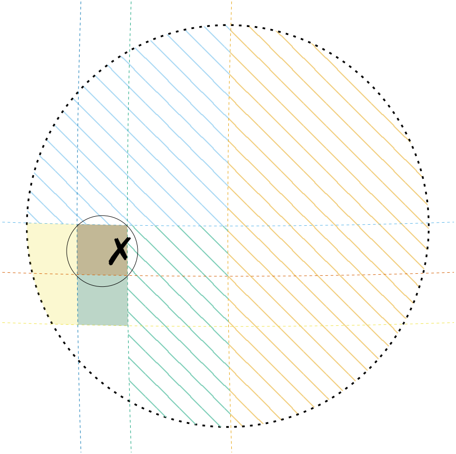


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing....
- Until ‘found’!
  (reduced to radius 1)
- Time?

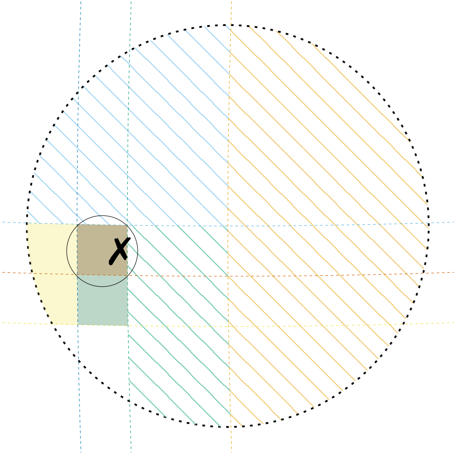


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\to \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time?
- Initial diameter? $2n$

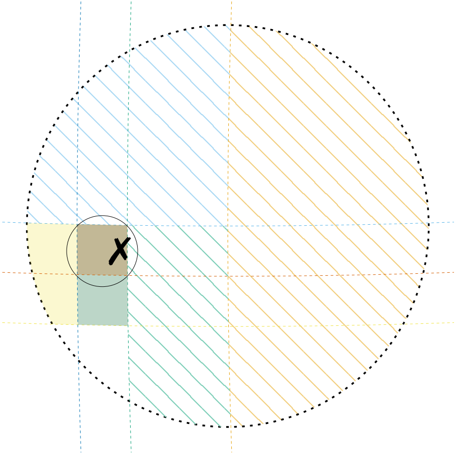


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time?
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$

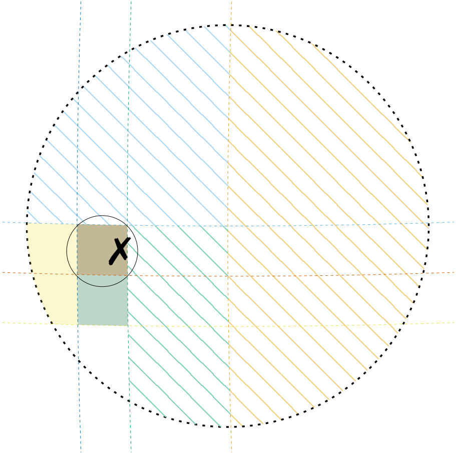


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\to \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$

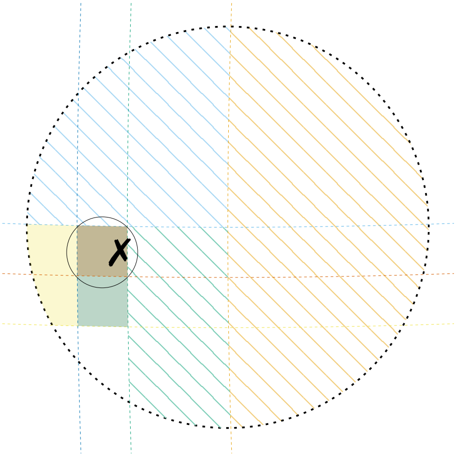


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\to \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal?

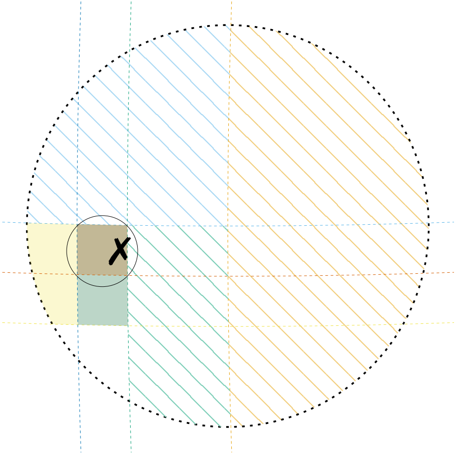


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\to \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal?
- Initial area: πn^2

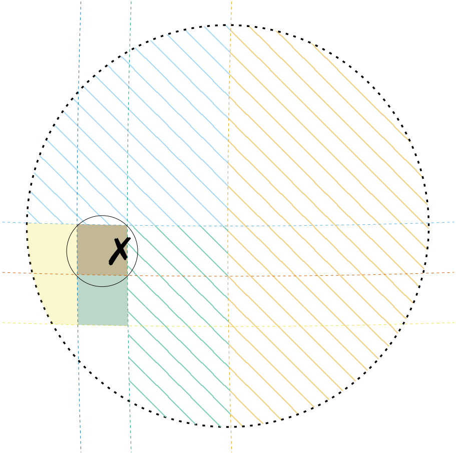


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal?
- Initial area: πn^2
- Final area: π

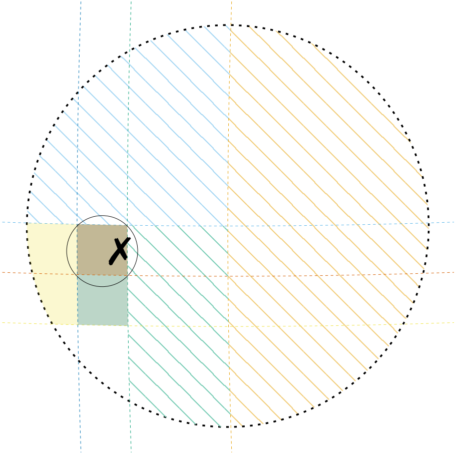


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius...... $\rightarrow \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal?
- Initial area: πn^2
- Final area: π
- Optimal probe $\rightarrow$ half remaining

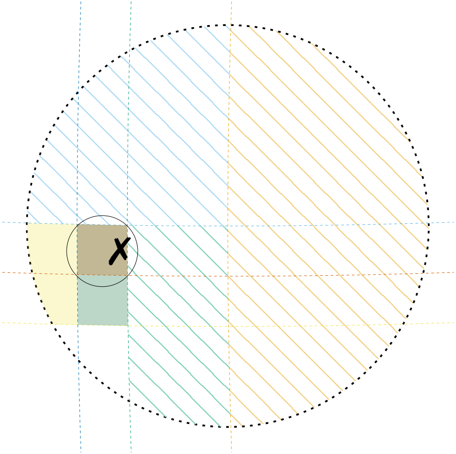


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\to \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal?
- Initial area: πn^2
- Final area: π
- Optimal probe $\to$ half remaining
- $\lceil \log(\pi n^2/\pi) \rceil = \lceil 2 \log n \rceil$

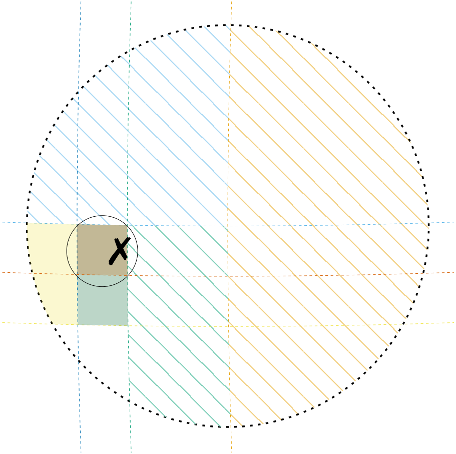


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗?
- Probe radius....... $\rightarrow \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal? pretty much!
- Initial area: πn^2
- Final area: π
- Optimal probe $\rightarrow$ half remaining
- $\lceil \log(\pi n^2 / \pi) \rceil = \lceil 2 \log n \rceil$

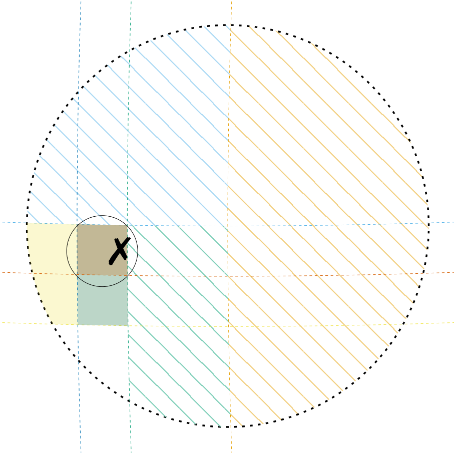


Figure 4: Trivial example w/ one ✗.

Why not One?

- What if just one ✗? too easy!
- Probe radius....... $\to \infty$
- Keep dividing....
- Until 'found'!
  (reduced to radius 1)
- Time? $2\lceil \log n \rceil + 2$
- Initial diameter? $2n$
- x halved $\log 2n = \lceil \log n \rceil + 1$
- Optimal? pretty much!
- Initial area: πn^2
- Final area: π
- Optimal probe $\to$ half remaining
- $\lceil \log(\pi n^2/\pi) \rceil = \lceil 2 \log n \rceil$

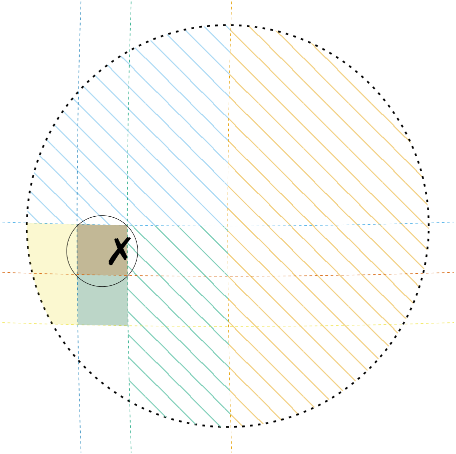


Figure 4: Trivial example w/ one ✗.

Why not One? Restrict Radius

- Limit radius to n?

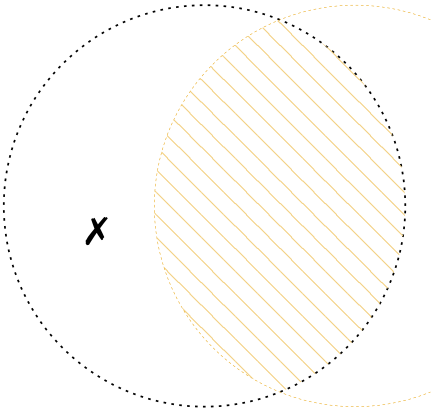


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide

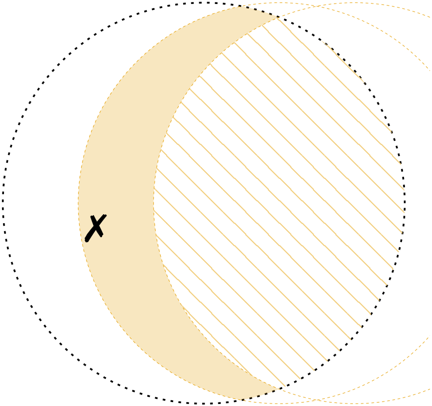


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide

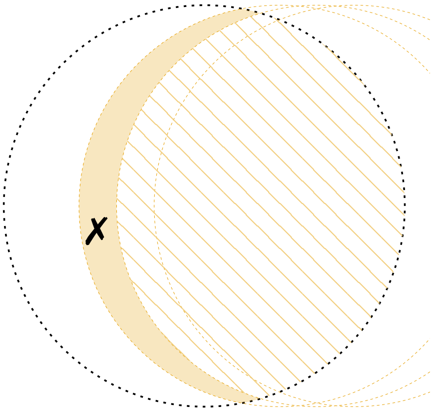


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$

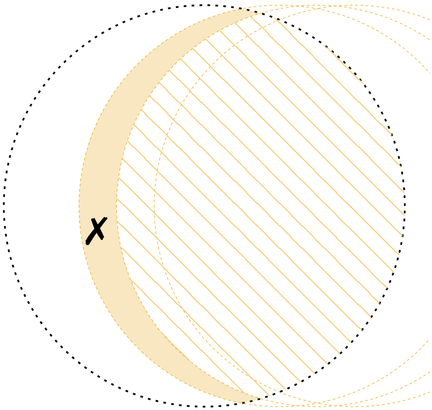


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide

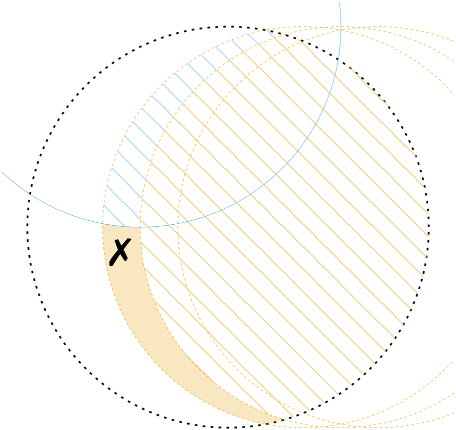


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide

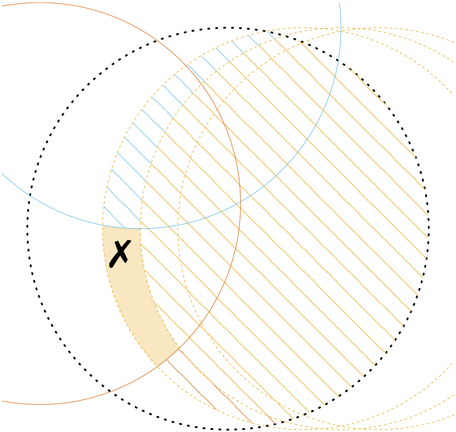


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide

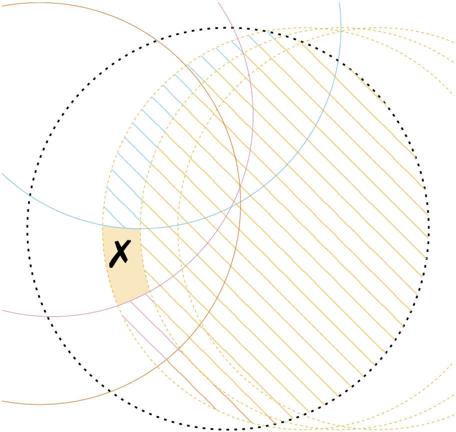


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide $\lceil \log \pi n \rceil$

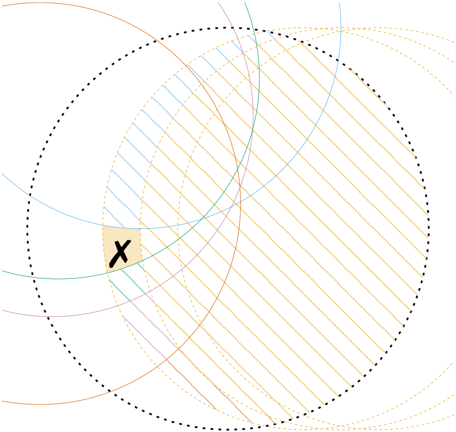


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide $\lceil \log \pi n \rceil$
- Overall: $\leq 2\lceil \log n \rceil + 3$

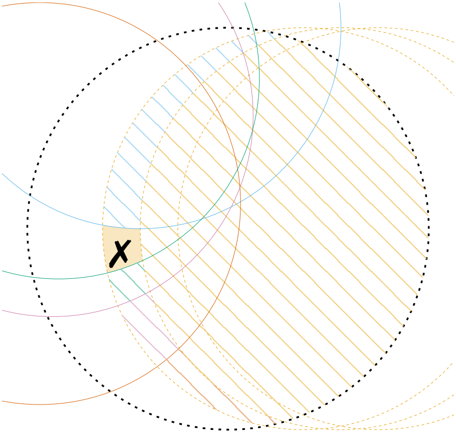


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide $\lceil \log \pi n \rceil$
- Overall: $\leq 2\lceil \log n \rceil + 3$
- Also close enough!

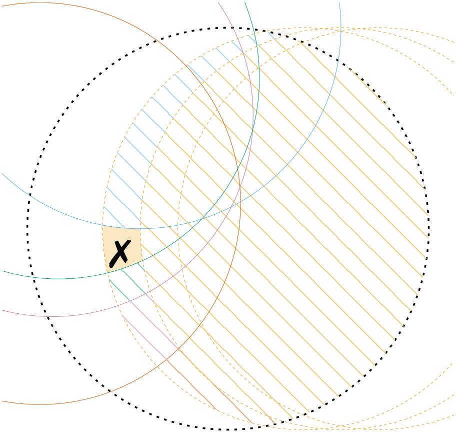


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?

- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide $\lceil \log \pi n \rceil$
- Overall: $\leq 2\lceil \log n \rceil + 3$
- Also close enough!

- Δ might leave initial area...

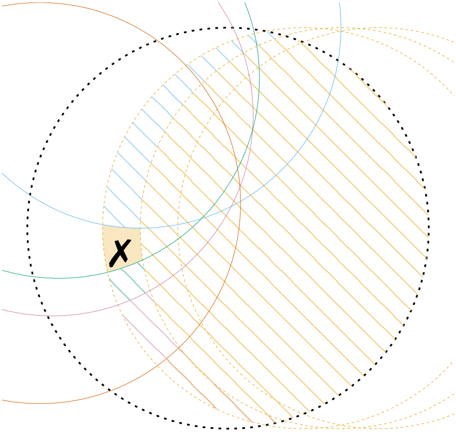


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?

- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide $\lceil \log \pi n \rceil$
- Overall: $\leq 2\lceil \log n \rceil + 3$
- Also close enough!

- Δ might leave initial area...
- Can also solve in $2\lceil \log n \rceil + \mathcal{O}(1)$

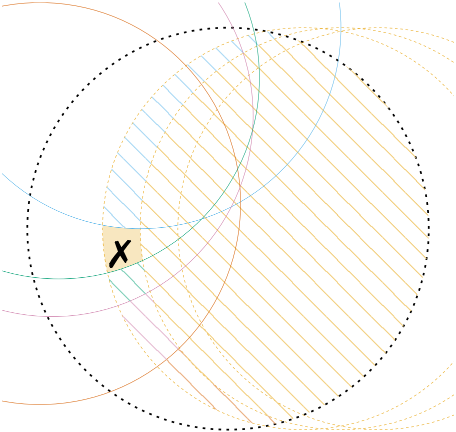


Figure 5: One ✗, probe $d \leq n$.

Why not One? Restrict Radius

- Limit radius to n?
- Restrict x to 1-wide $\lceil \log 2n \rceil$
- Restrict y to 1-wide $\lceil \log \pi n \rceil$
- Overall: $\leq 2\lceil \log n \rceil + 3$
- Also close enough!
- Δ might leave initial area...
- Can also solve in $2\lceil \log n \rceil + \mathcal{O}(1)$
- From now on...
  1. probe radius $\leq n$
  2. may be multiple ✗!

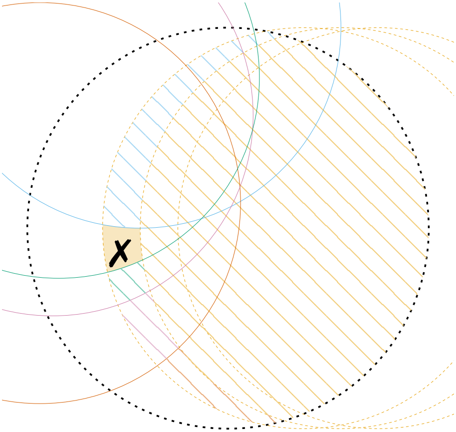


Figure 5: One ✗, probe $d \leq n$.

L_2: Hexagonal Algorithms

- Consider hexagonal lattice

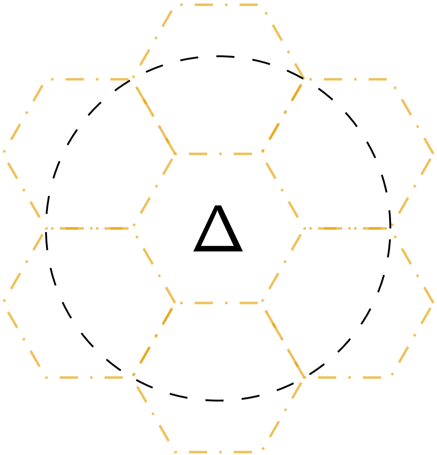


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$

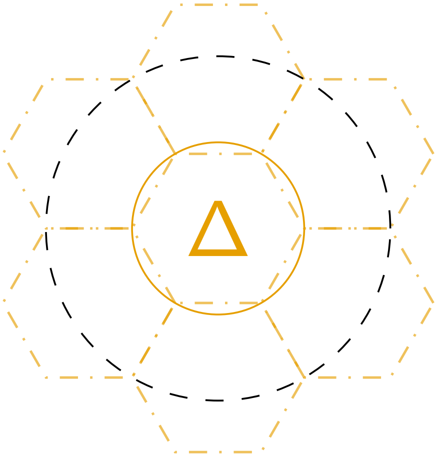


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!

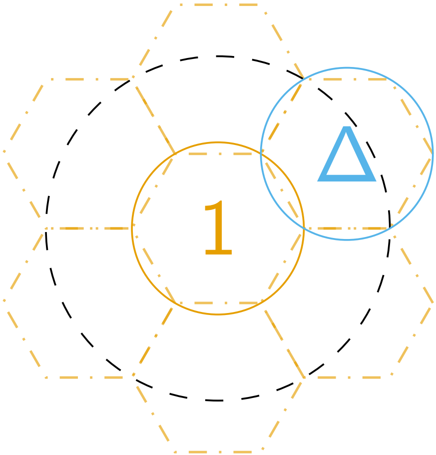


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!

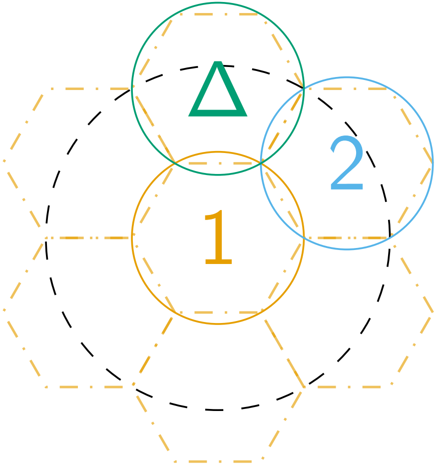


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!

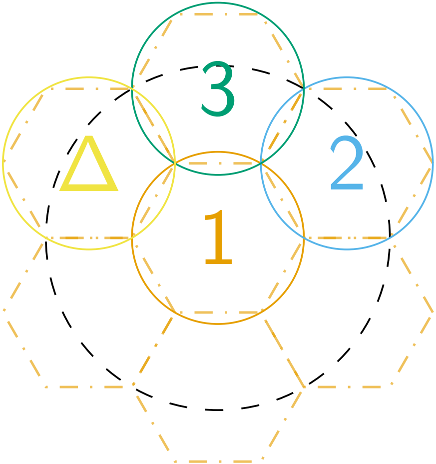


Figure 6: Algorithm 1

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!

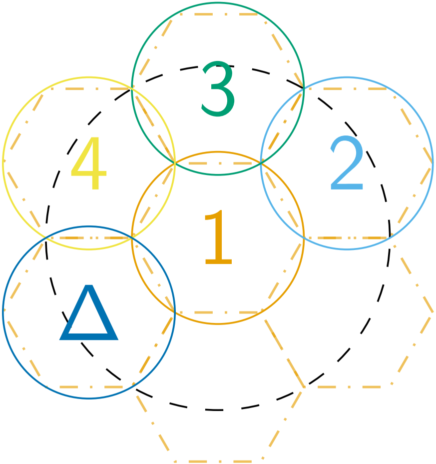


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!

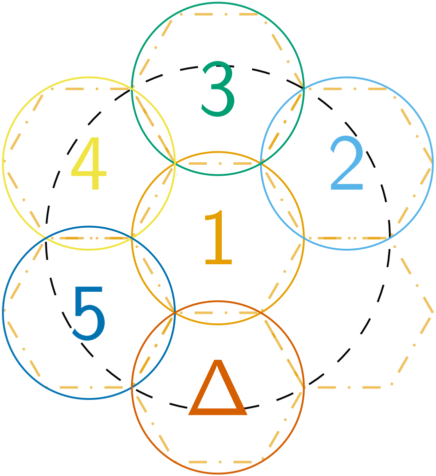


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...

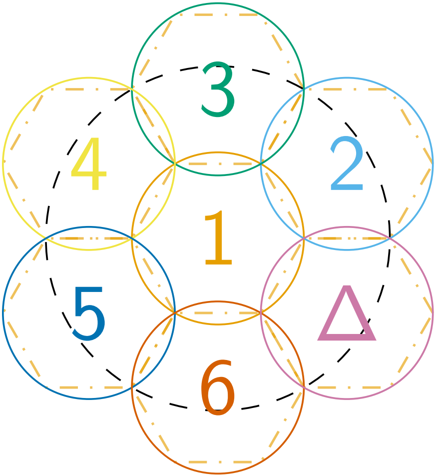


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!

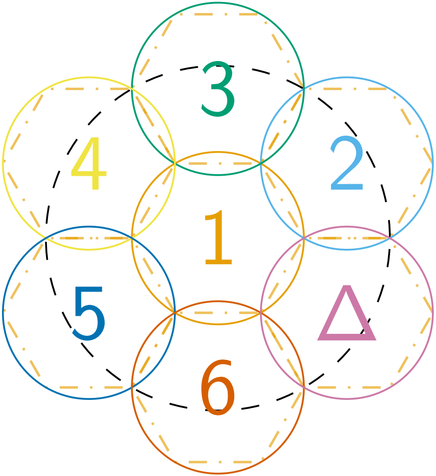


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!
- Recurse!
- $P(n) \leq 6\lceil \log n \rceil$

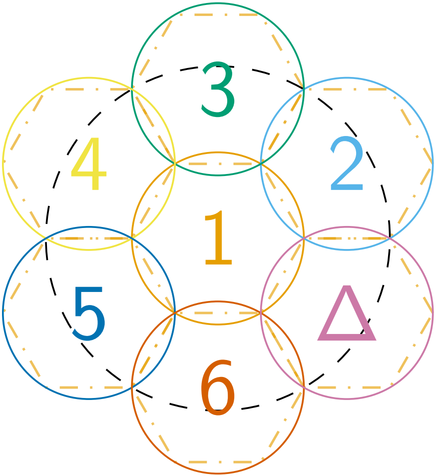


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!
- Recurse!
- $P(n) \leq 6\lceil \log n \rceil$
- Total responses?

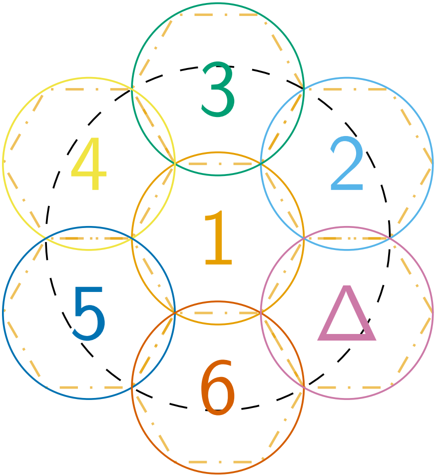


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!
- Recurse!
- $P(n) \leq 6\lceil \log n \rceil$
- Total responses?
- At most one per layer...

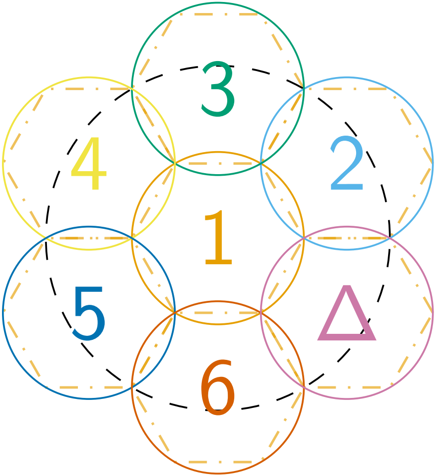


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!
- Recurse!
- $P(n) \leq 6\lceil \log n \rceil$
- Total responses?
- At most one per layer...
- $R(n) \leq \lceil \log n \rceil$

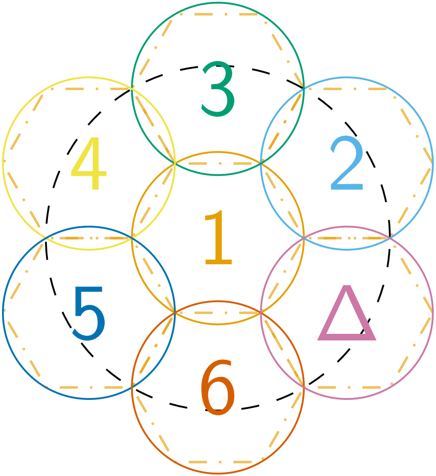


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!
- Recurse!
- $P(n) \leq 6\lceil \log n \rceil$
- Total responses?
- At most one per layer...
- $R(n) \leq \lceil \log n \rceil$
- Distance traveled?

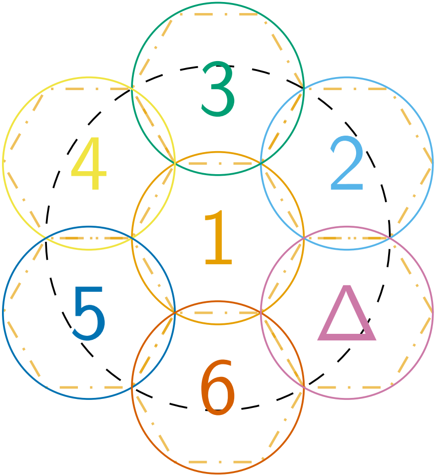


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms

- Consider hexagonal lattice
- Hexagon side length: $n/2$
- Probe!
- After (at most) 6 probes...
- Search area radius is halved!
- Recurse!
- $P(n) \leq 6\lceil \log n \rceil$
- Total responses?
- At most one per layer...
- $R(n) \leq \lceil \log n \rceil$
- Distance traveled?
- $D(n) \leq 10.39n$

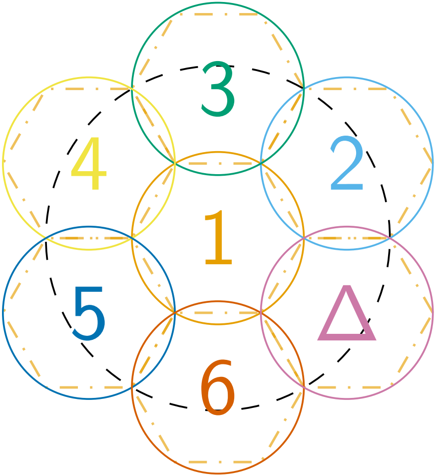


Figure 6: Algorithm 1

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice

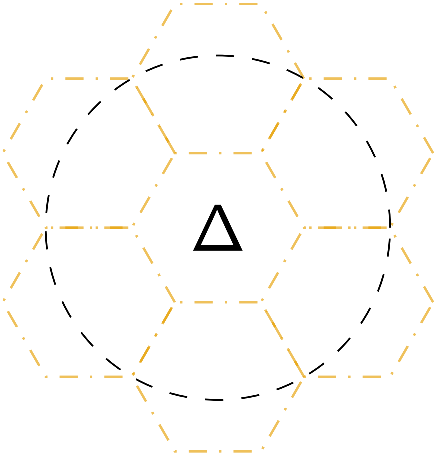


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...

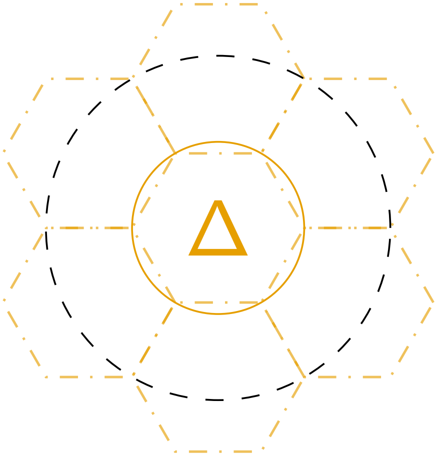


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...

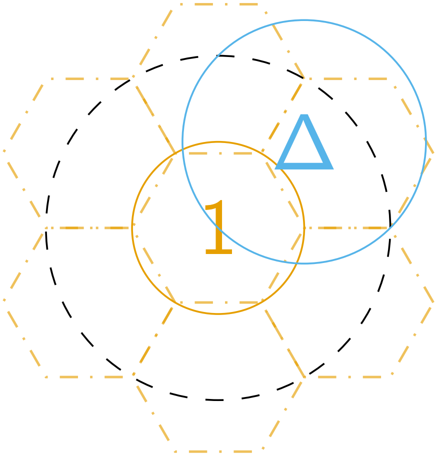


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...

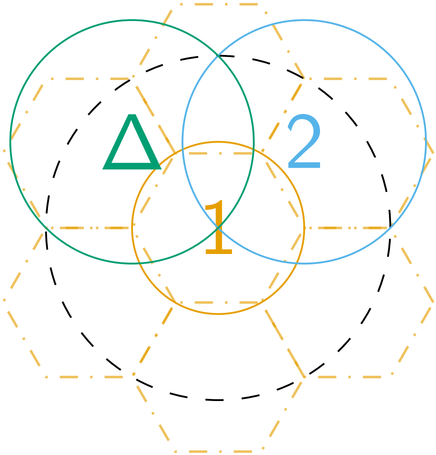


Figure 7: Algorithm 2

- Consider hexagonal lattice
- First probe 2 quadrants...

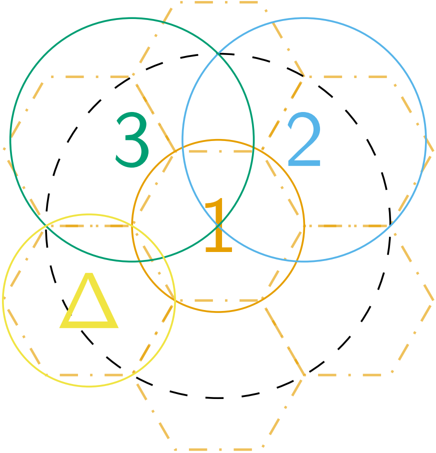


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...

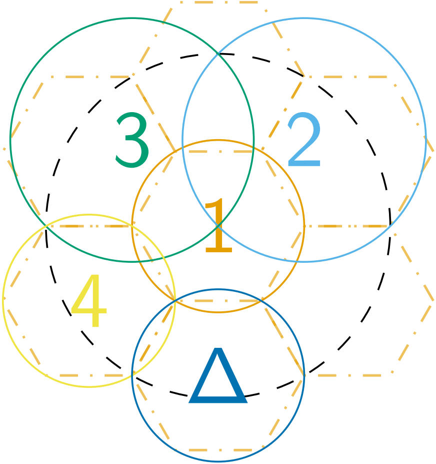


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!

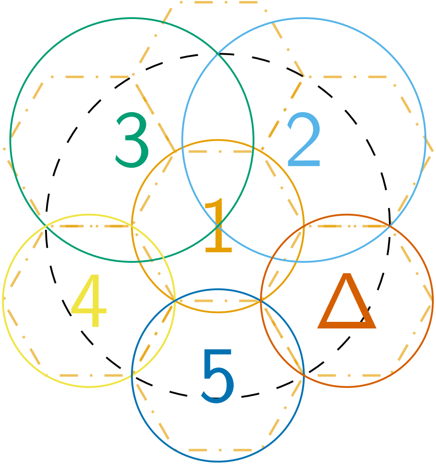


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!
- $P(n) \leq 5\lceil \log n \rceil$

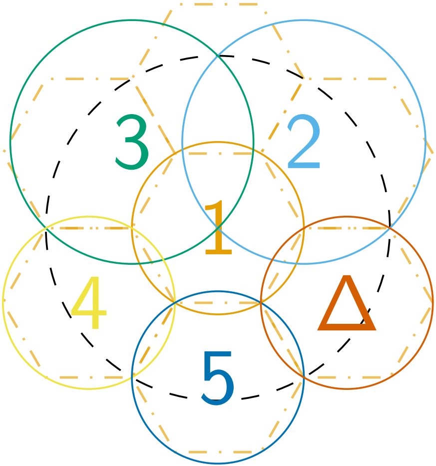


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!
- $P(n) \leq 5\lceil \log n \rceil$
- Total responses?

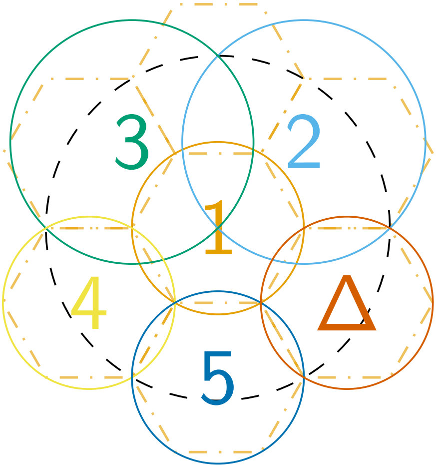


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!
- $P(n) \leq 5\lceil \log n \rceil$
- Total responses?
- If 3rd probe succeeds, only $1/\sqrt{2}$ reduction...

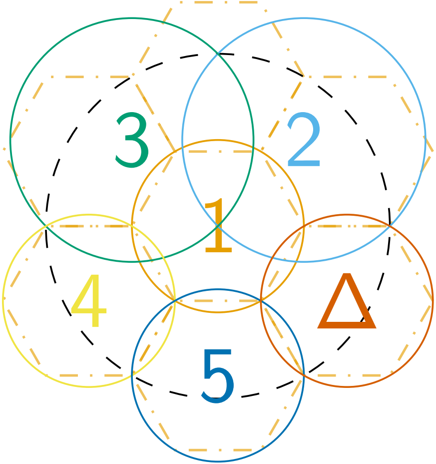


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!
- $P(n) \leq 5\lceil \log n \rceil$
- Total responses?
- If 3rd probe succeeds, only $1/\sqrt{2}$ reduction...
- $R(n) \leq 2\lceil \log n \rceil$

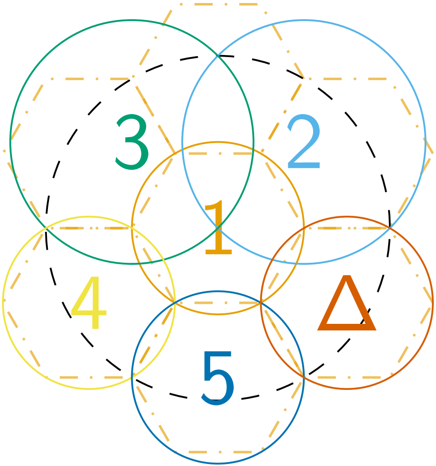


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!
- $P(n) \leq 5\lceil \log n \rceil$
- Total responses?
- If 3rd probe succeeds, only $1/\sqrt{2}$ reduction...
- $R(n) \leq 2\lceil \log n \rceil$
- Distance traveled?

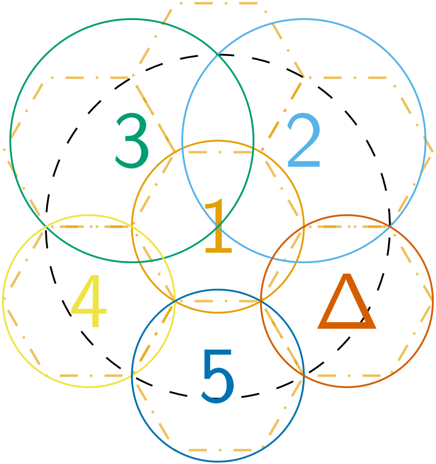


Figure 7: Algorithm 2

L_2: Hexagonal Algorithms cont.

- Consider hexagonal lattice
- First probe 2 quadrants...
- After (at most) 5 probes...
- Search area radius is halved!
- $P(n) \leq 5\lceil \log n \rceil$
- Total responses?
- If 3rd probe succeeds, only $1/\sqrt{2}$ reduction...
- $R(n) \leq 2\lceil \log n \rceil$
- Distance traveled?
- $D(n) \leq 8.81n$

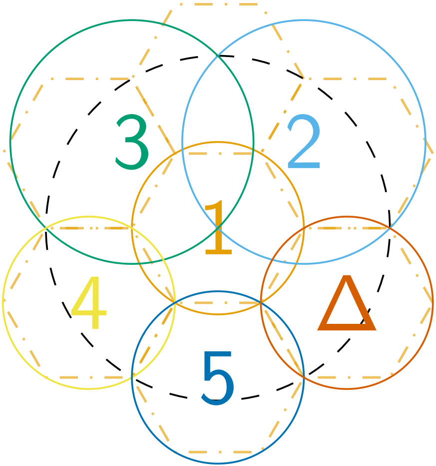


Figure 7: Algorithm 2

L_2: Progressively Shrinking Probes

- Larger probes are better?

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed

- Best size?

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed

- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!
- How much?

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!
- How much?
- Let k-th probe have $r_k = \rho_k n$

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed

- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!

- How much?
- Let k-th probe have $r_k = \rho_k n$
- Solve recurrence: $P(n) = k + P(\rho_k n)$

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!
- How much?
- Let k-th probe have $r_k = \rho_k n$
- Solve recurrence: $P(n) = k + P(\rho_k n)$
- Minimum when $\rho_k = \rho_1^k$

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed

- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!

- How much?
- Let k-th probe have $r_k = \rho_k n$
- Solve recurrence: $P(n) = k + P(\rho_k n)$
- Minimum when $\rho_k = \rho_1^k$
- Geometrically decreasing!

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!
- How much?
- Let k-th probe have $r_k = \rho_k n$
- Solve recurrence:
  $P(n) = k + P(\rho_k n)$
- Minimum when $\rho_k = \rho_1^k$
- Geometrically decreasing!

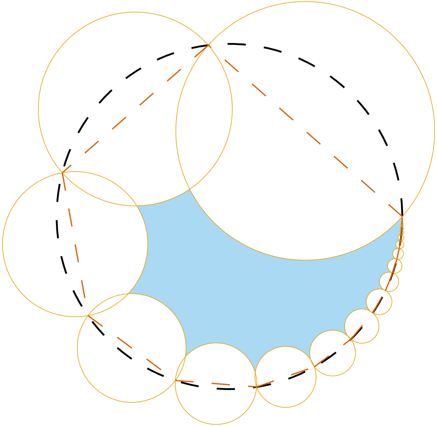


Figure 8: Must be able to cover perimeter... $\rho_1 \geq 0.74915\ldots$

L_2: Progressively Shrinking Probes

- Larger probes are better?
  - ☑ Reduce more when fail
  - ☒ Reduce less when succeed
- Best size? depends when probed
- Intuition: If spent more probes, should expect better reduction
- Probes progressively shrink!
- How much?
- Let k-th probe have $r_k = \rho_k n$
- Solve recurrence:
  $P(n) = k + P(\rho_k n)$
- Minimum when $\rho_k = \rho_1^k$
- Geometrically decreasing!

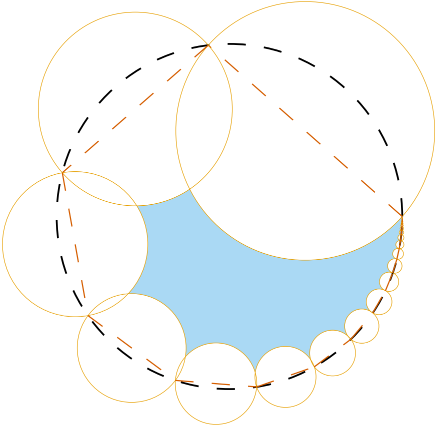


Figure 8: Must be able to cover perimeter... $\rho_1 \geq 0.74915\ldots$

Lower bound: $P(n) \geq 2.40001\lceil \log n \rceil$

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small

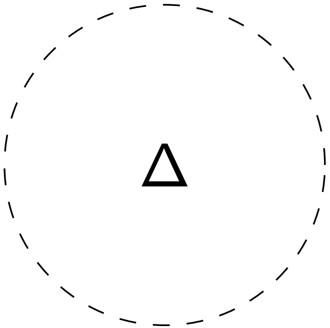


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

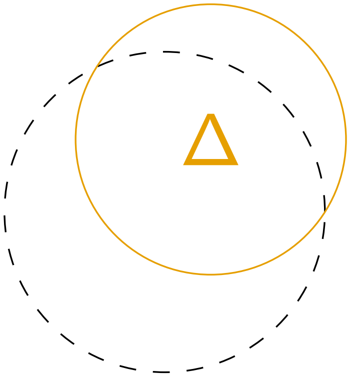


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

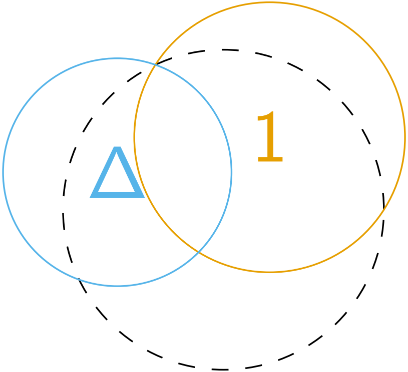


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

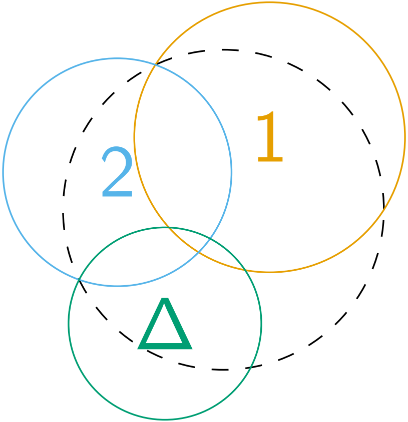


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

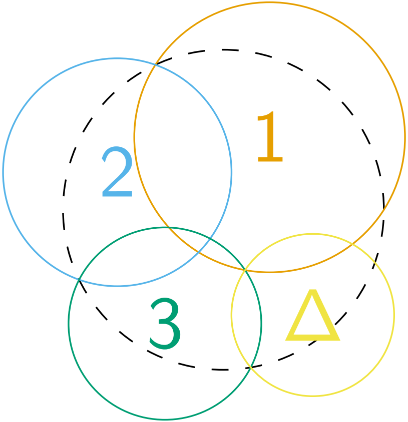


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

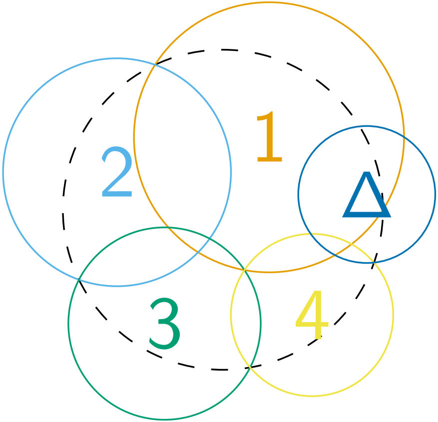


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$

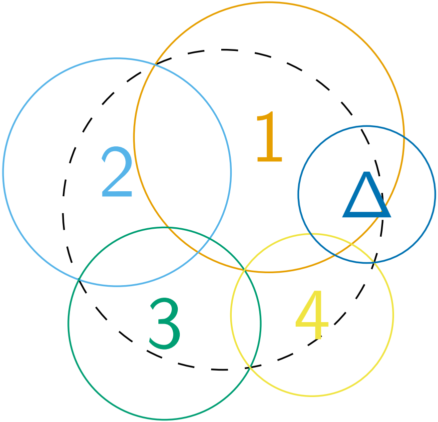


Figure 9: Algorithm 3

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$

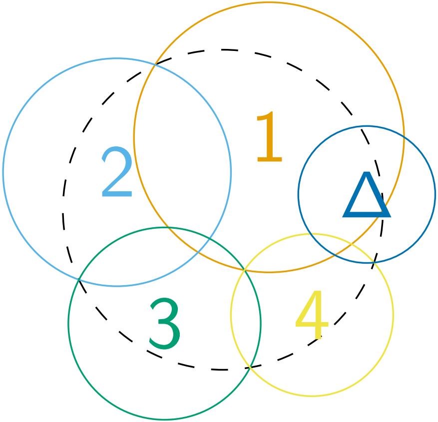


Figure 9: Algorithm 3

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

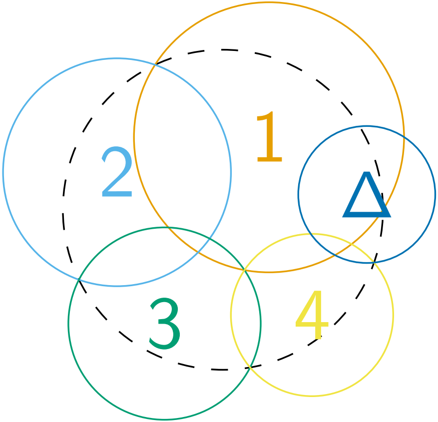


Figure 9: Algorithm 3

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \le -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \le 6.95n$

- What if we rearrange?

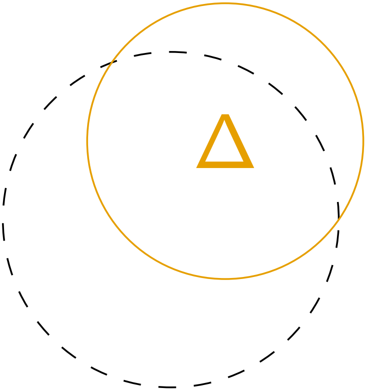


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?

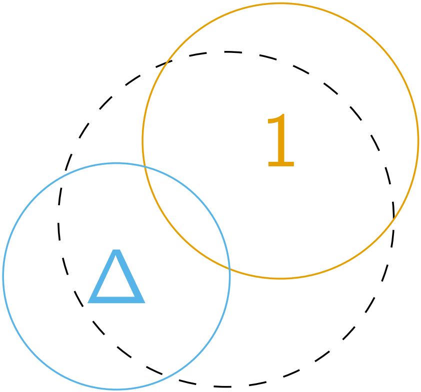


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?

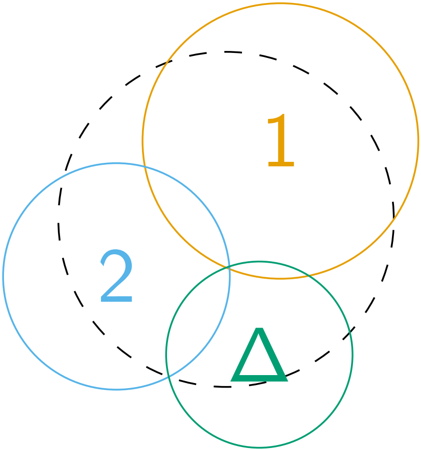


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil =$ $4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?

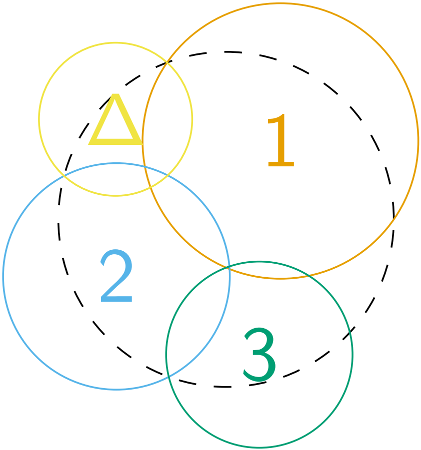


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?
- Can get $\rho_1 = 0.822\ldots$

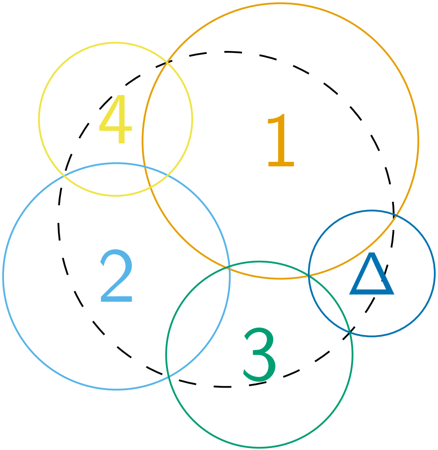


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?
- Can get $\rho_1 = 0.822\ldots$
- $P(n) = R(n) \leq 3.54\lceil \log n \rceil$

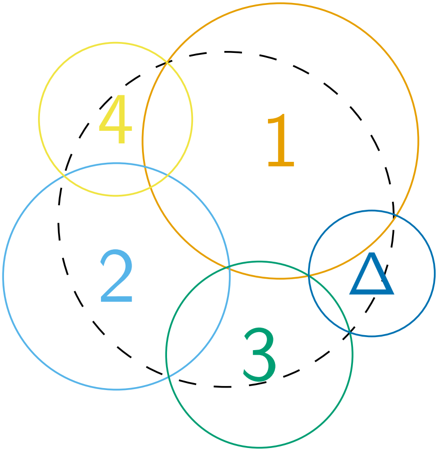


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil = 4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?
- Can get $\rho_1 = 0.822\ldots$
- $P(n) = R(n) \leq 3.54\lceil \log n \rceil$
- $D(n) \leq 9.31n$

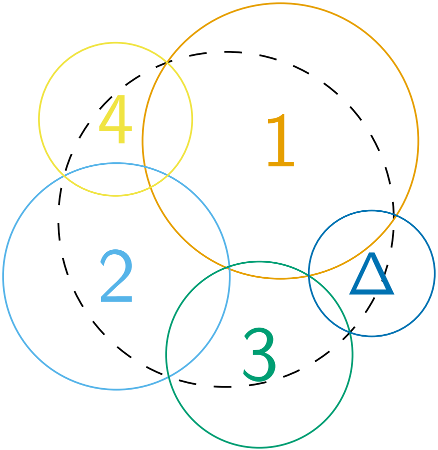


Figure 9: Algorithm 4

L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil =$ $4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?
- Can get $\rho_1 = 0.822\ldots$
- $P(n) = R(n) \leq 3.54\lceil \log n \rceil$
- $D(n) \leq 9.31n$

- Why only 5 probes?

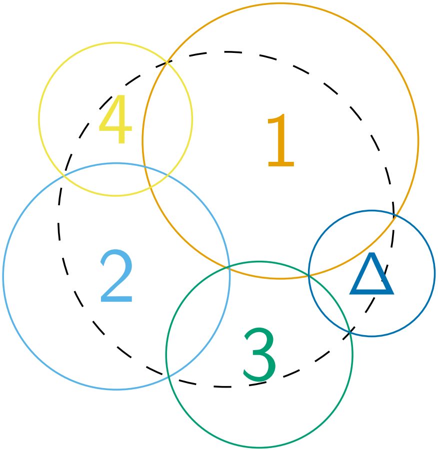


Figure 9: Algorithm 4

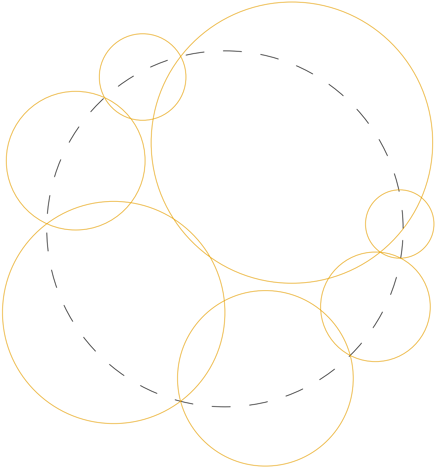
L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\dots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\dots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil =$ $4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?
- Can get $\rho_1 = 0.822\dots$
- $P(n) = R(n) \leq 3.54\lceil \log n \rceil$
- $D(n) \leq 9.31n$

- Why only 5 probes? Smaller probes $\rightarrow$ uncovered center



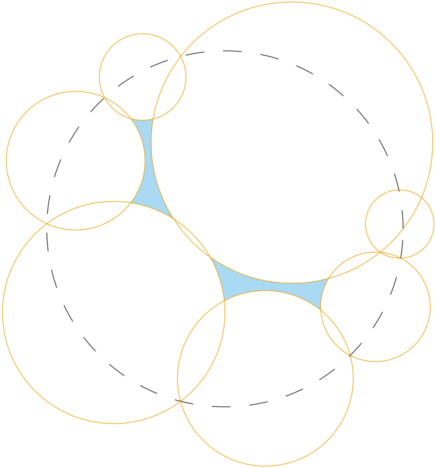
L_2: Chord-Based Shrinking Algorithms

- $\rho_1 = 0.74915\ldots$ too small
- How large must ρ_1 be?

- $\rho_1 = 0.844\ldots$
- $P(n) = R(n) \leq -\frac{1}{\log \rho_1}\lceil \log n \rceil =$ $4.08\lceil \log n \rceil$
- $D(n) \leq 6.95n$

- What if we rearrange?
- Can get $\rho_1 = 0.822\ldots$
- $P(n) = R(n) \leq 3.54\lceil \log n \rceil$
- $D(n) \leq 9.31n$

- Why only 5 probes? Smaller probes $\rightarrow$ uncovered center



L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...

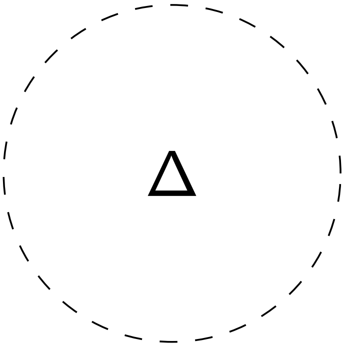


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...

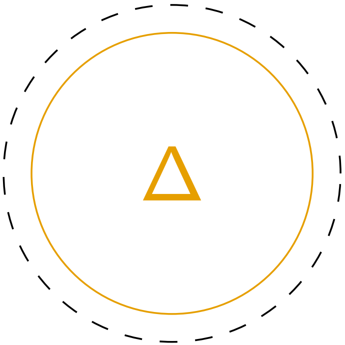


Figure 9: Algorithm 5

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.

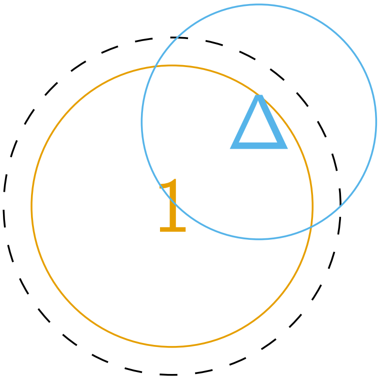


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter..

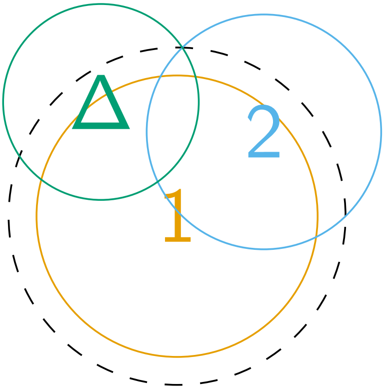


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter...

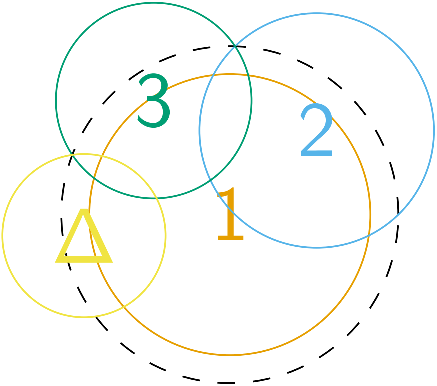


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter....

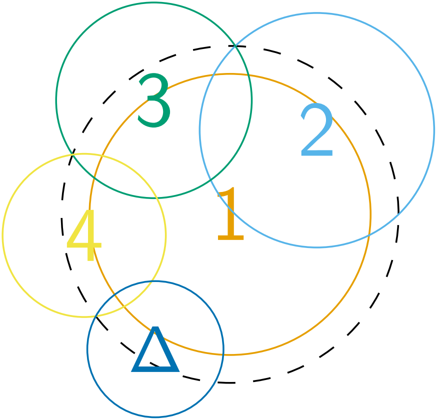


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.....

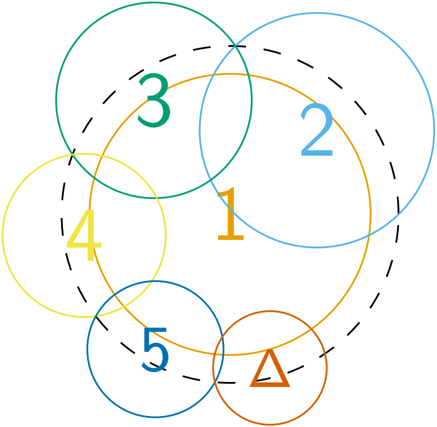


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter......

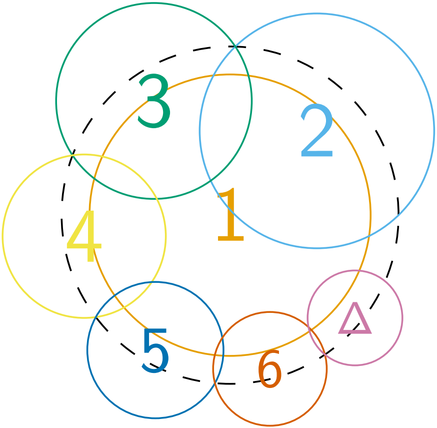


Figure 9: Algorithm 5

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter.......
- 7 probes!

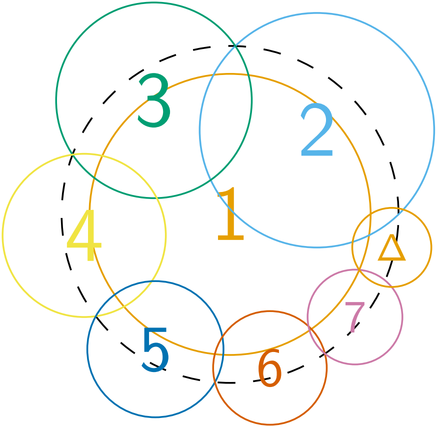


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......

- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$

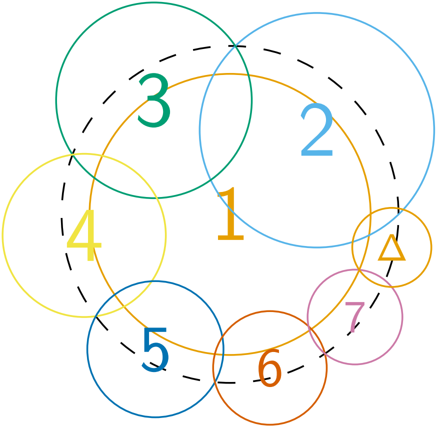


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......

- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$

- Problem: Outer circumference covered faster than inner

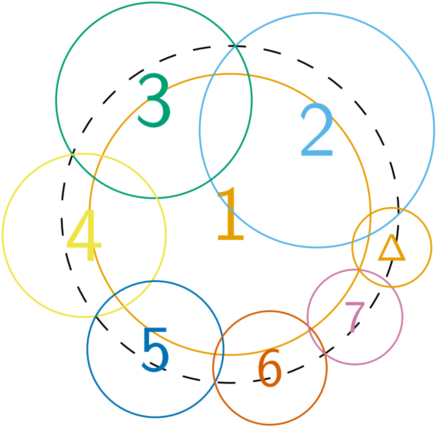


Figure 9: Algorithm 5

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83 \lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate

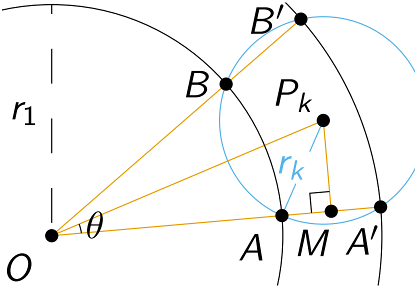


Figure 9: Some geometry

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter.......
- 7 probes! but inefficient. . .
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

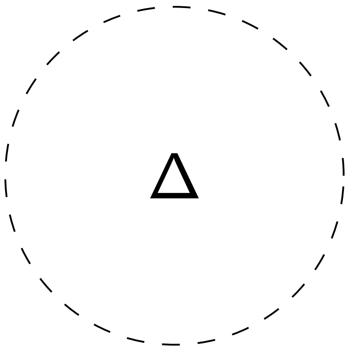


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

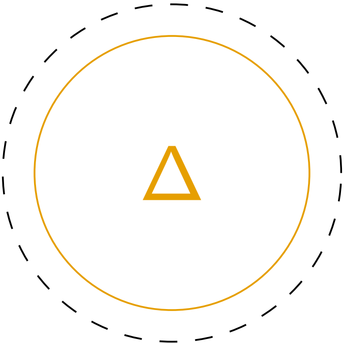


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

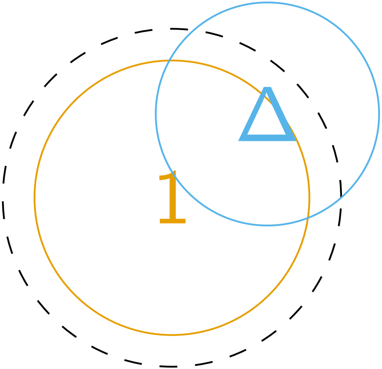


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

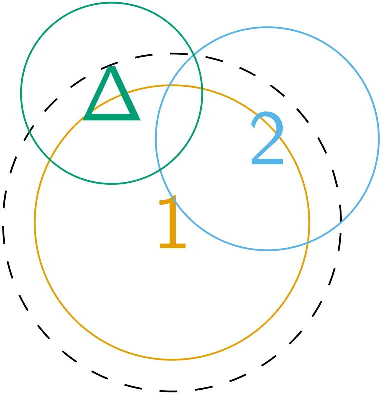


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

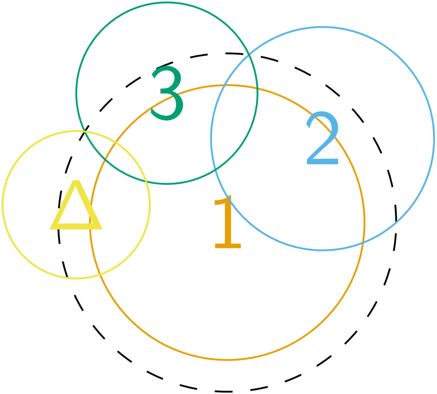


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

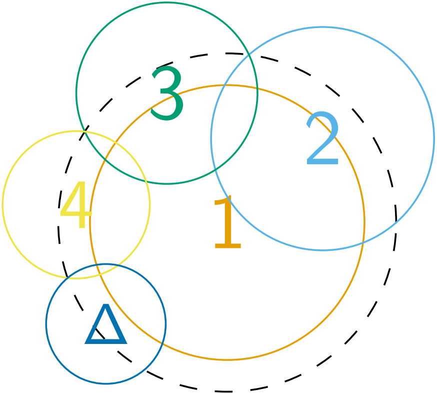


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter.......

- 7 probes! but inefficient. . .
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$

- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate

- Result:

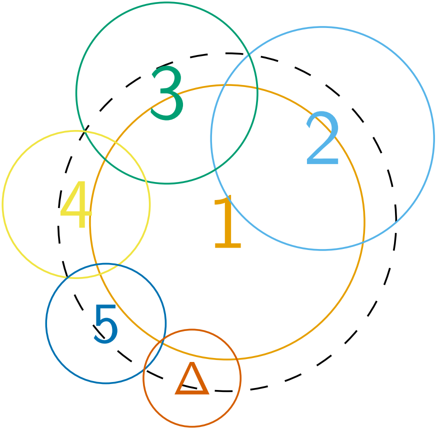


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter.......

- 7 probes! but inefficient. . .
- $P(n) \leq 3.83 \lceil \log n \rceil$
- $D(n) \leq 6.72n$

- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate

- Result:

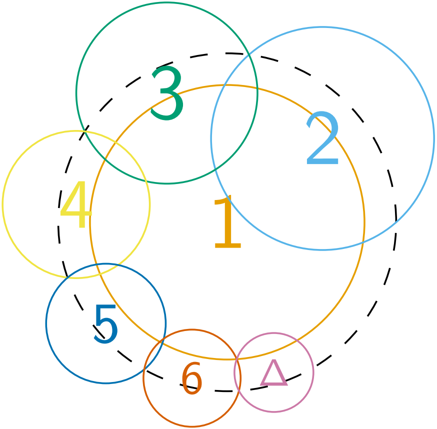


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter.......
- 7 probes! but inefficient. . .
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

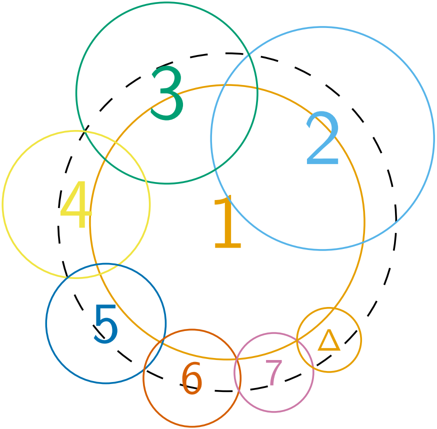


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......

- 7 probes! but inefficient...
- $P(n) \leq 3.83 \lceil \log n \rceil$
- $D(n) \leq 6.72 n$

- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate

- Result:

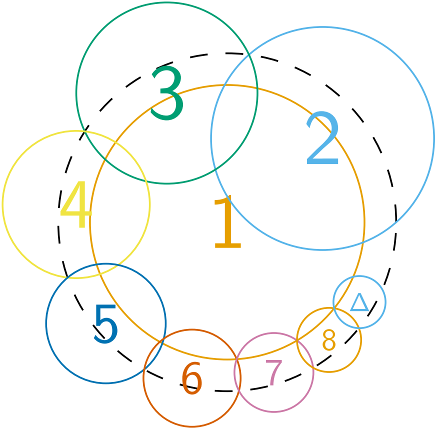


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

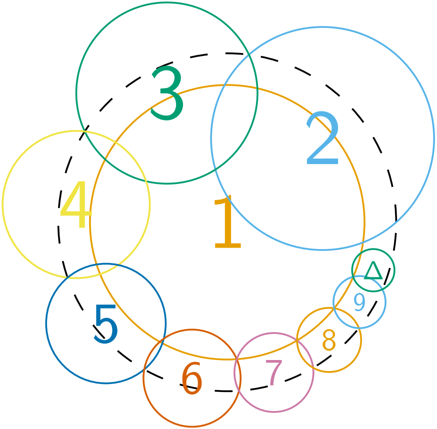


Figure 9: Algorithm 6

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result:

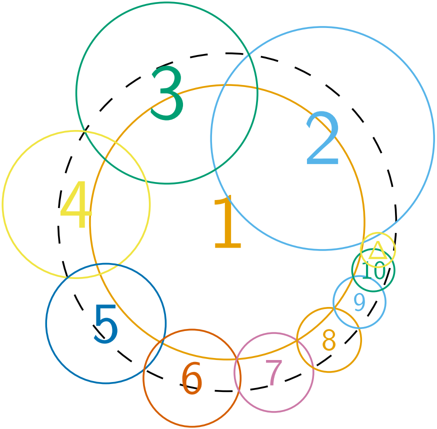


Figure 9: Algorithm 6

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......
- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$
- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate
- Result: turned up to 11!

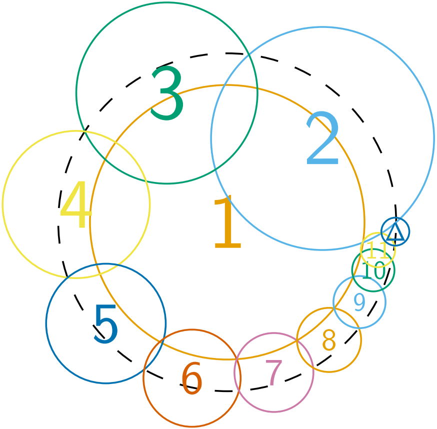


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center...
- Let's start in center...
- Then perimeter.......

- 7 probes! but inefficient...
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$

- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate

- Result: turned up to 11!
- $P(n) \leq 3.34\lceil \log n \rceil$
- $D(n) \leq 6.02n$

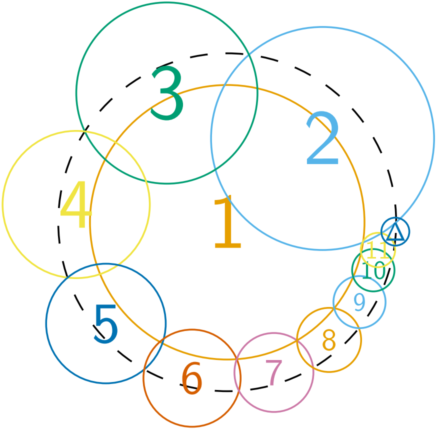


Figure 9: Algorithm 6

L_2: Higher-count Monotonic-path Algorithms

- Avoid uncovered center. . .
- Let's start in center. . .
- Then perimeter.......

- 7 probes! but inefficient. . .
- $P(n) \leq 3.83\lceil \log n \rceil$
- $D(n) \leq 6.72n$

- Problem: Outer circumference covered faster than inner
- Solution: Cover at same rate

- Result: turned up to 11!
- $P(n) \leq 3.34\lceil \log n \rceil$
- $D(n) \leq 6.02n$ – our best result!

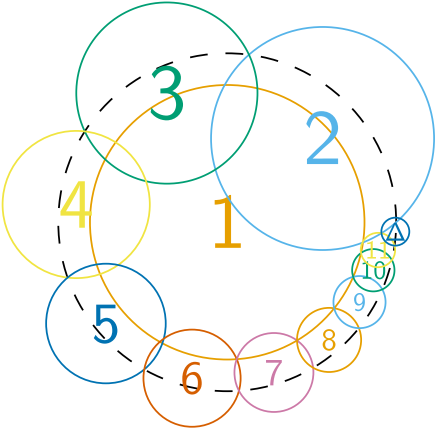
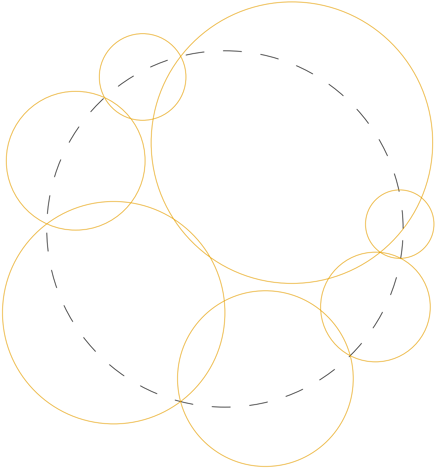


Figure 9: Algorithm 6

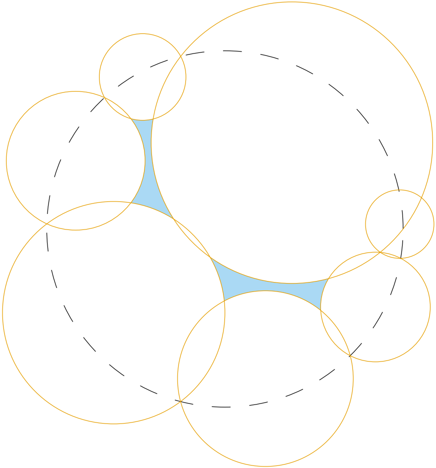
L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...



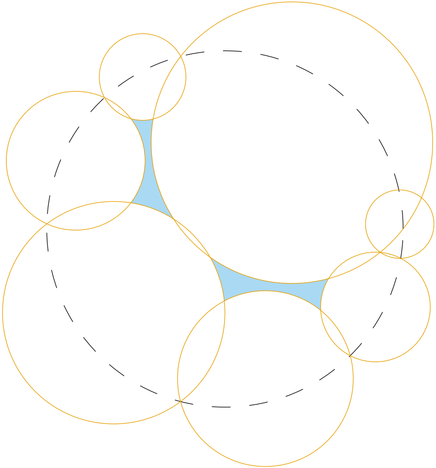
L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗



L_2: Darting Non-Monotonic Algorithms

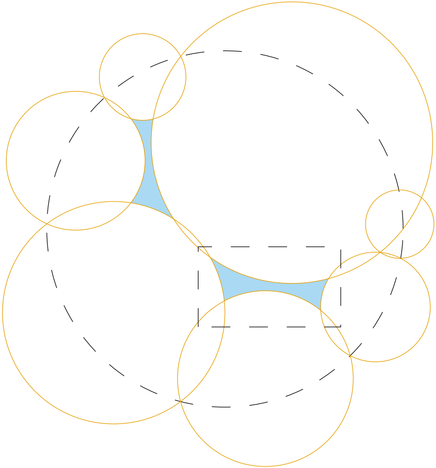
- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...



L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small…
- Uncovered internal area ✗
- Can add more probes…

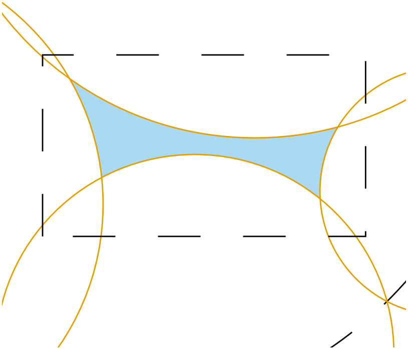
- How?



L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

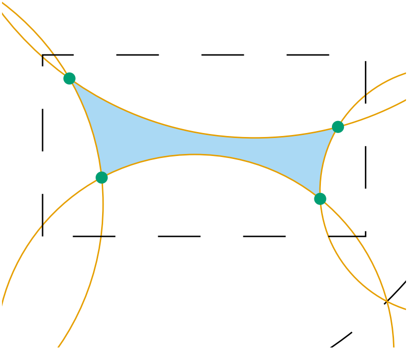
- How?



L_2: Darting Non-Monotonic Algorithms

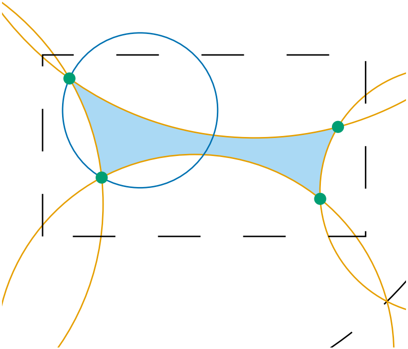
- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...



- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

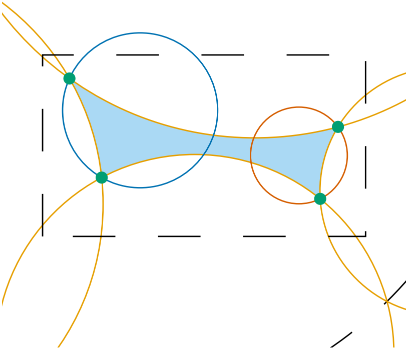
- How?
  1. Identify corners...
  2. Maximize coverage.



L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage..

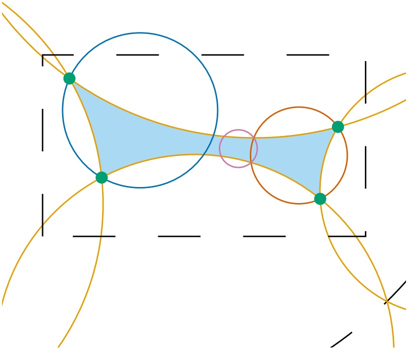


L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!



L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...
- How?
  1. Identify corners...
  2. Maximize coverage...
- Fill in holes programmatically!
- Start w/ Algorithm 4 ...

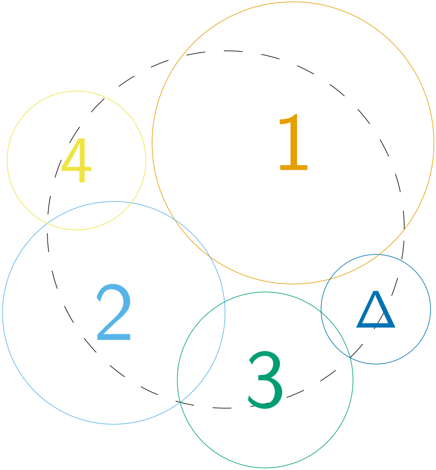


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically.

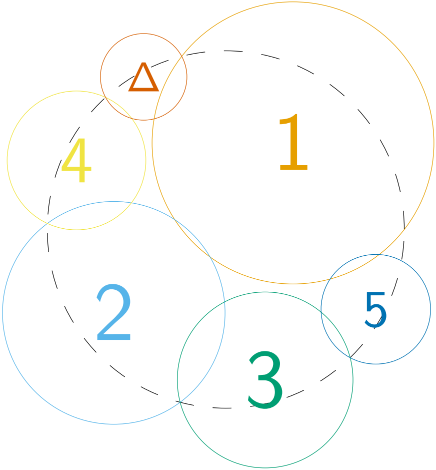


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically..

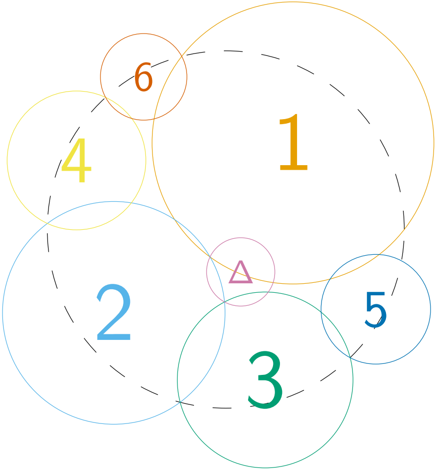


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically...

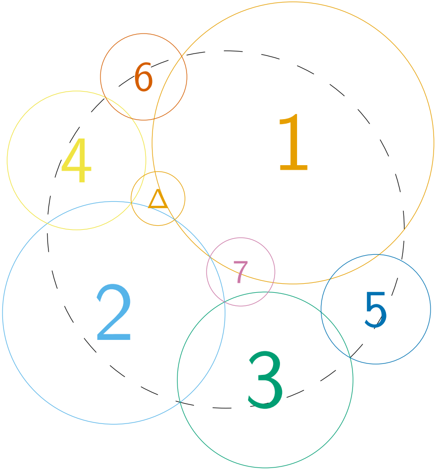


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically....

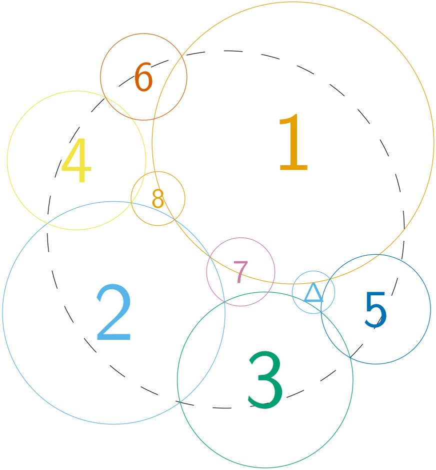


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...
- How?
  1. Identify corners...
  2. Maximize coverage...
- Fill in holes programmatically!
- Start w/ Algorithm 4 ...
- Finish programmatically.....

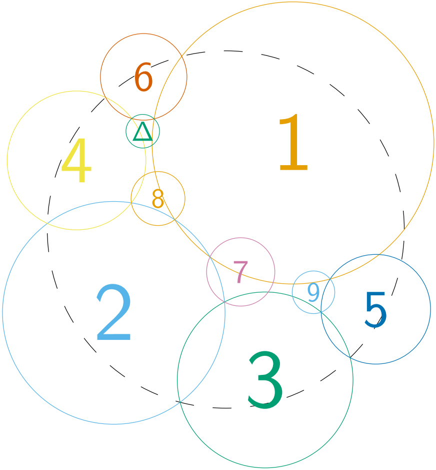


Figure 10: Algorithm 7

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically......

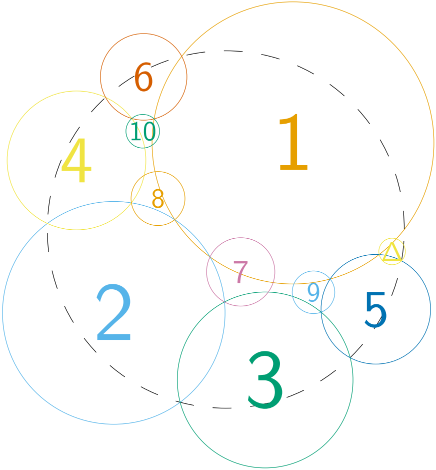


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically.......

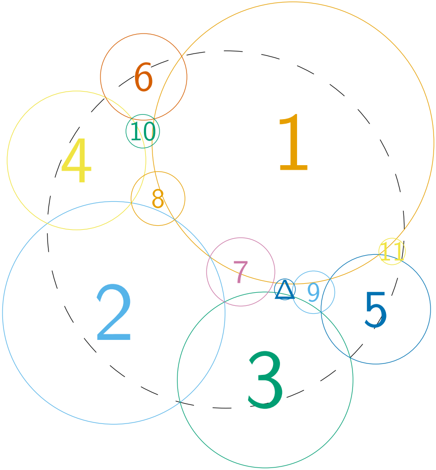


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically........

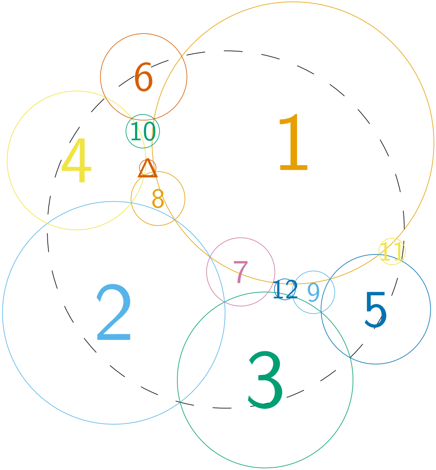


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically........

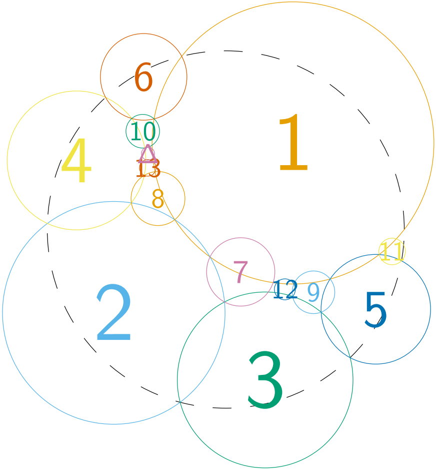


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically.........

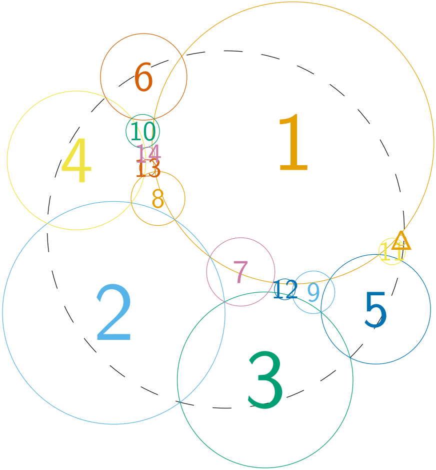


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small. . .
- Uncovered internal area ✗
- Can add more probes. . .

- How?
  1. Identify corners. . .
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 . . .
- Finish programmatically..........

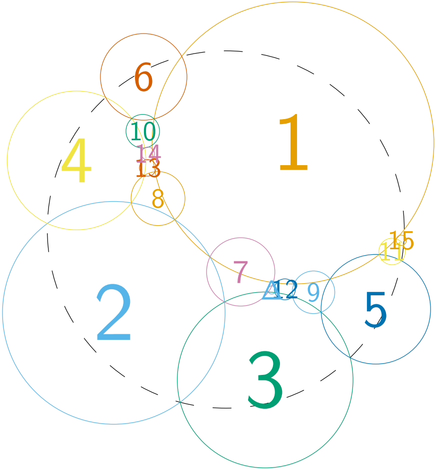


Figure 10: Algorithm 7

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically...........
- 25 probes!

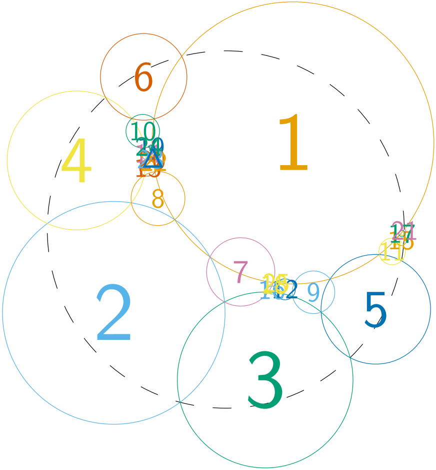


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically...........
- 25 probes!
- $P(n) \leq 2.93\lceil \log n \rceil$

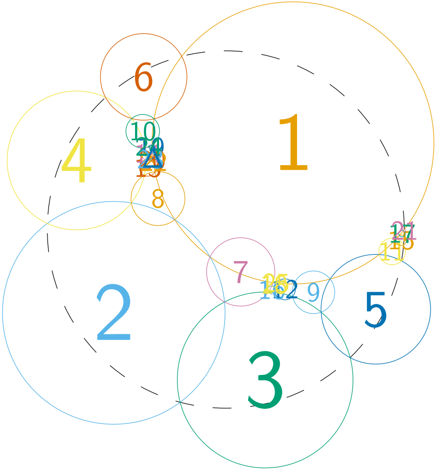


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically...........
- 25 probes!
- $P(n) \leq$ 2.93$\lceil \log n \rceil$
- $D(n) \leq$ 25.8n

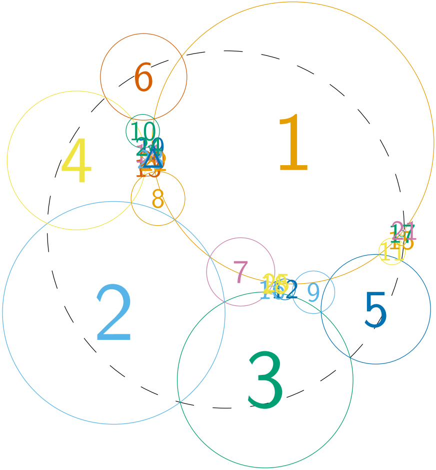


Figure 10: Algorithm 7

L_2: Darting Non-Monotonic Algorithms

- Algorithm 4 w/ ρ_1 too small...
- Uncovered internal area ✗
- Can add more probes...

- How?
  1. Identify corners...
  2. Maximize coverage...

- Fill in holes programmatically!

- Start w/ Algorithm 4 ...
- Finish programmatically...........
- 25 probes!
- $P(n) \leq 2.93\lceil \log n \rceil$
- $D(n) \leq 25.8n$ – terrible!

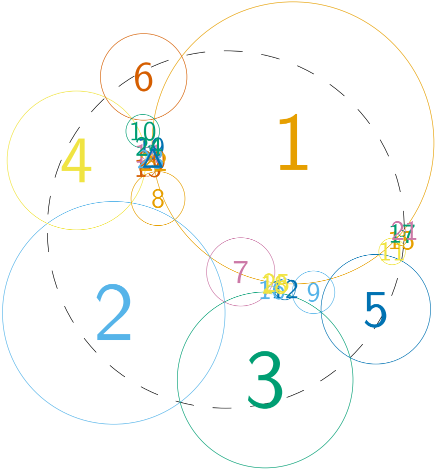
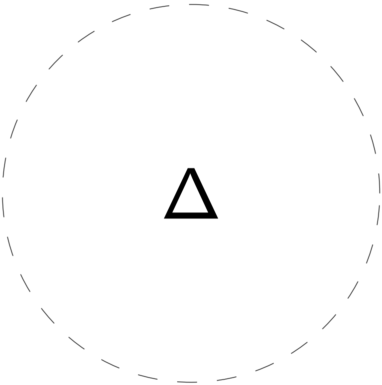


Figure 10: Algorithm 7

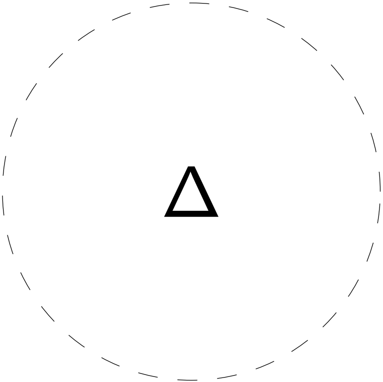
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?



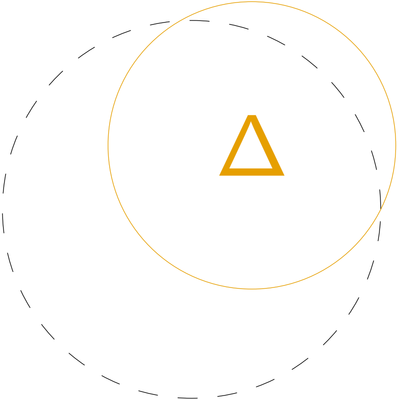
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop



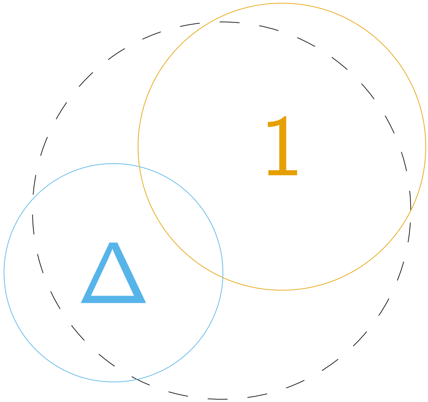
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution.



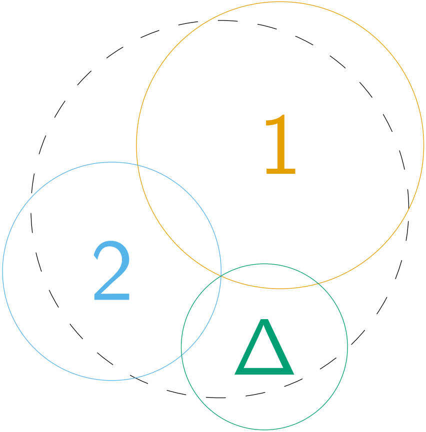
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution..



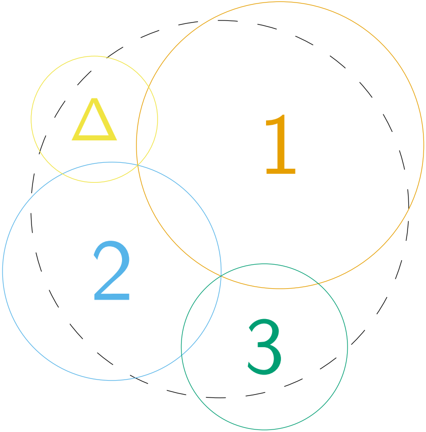
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution...

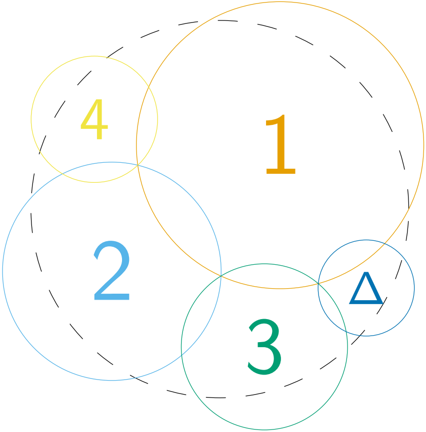


L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution....

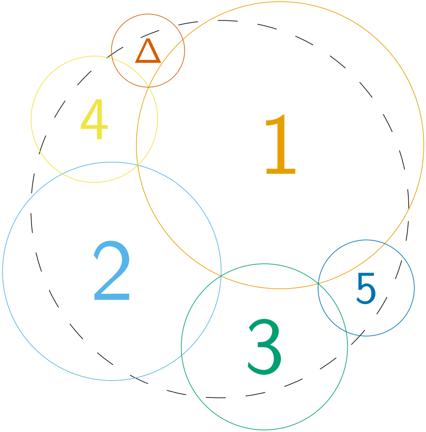


- How far can we push this?
- Take human out of the loop
- Differential evolution.....



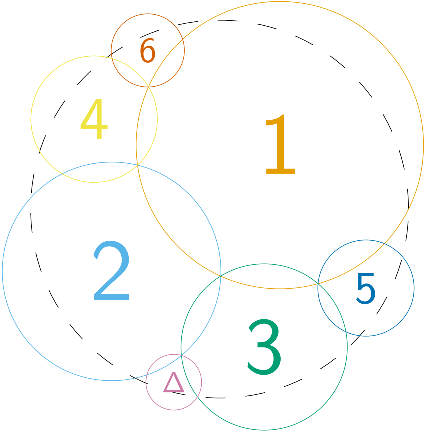
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution......



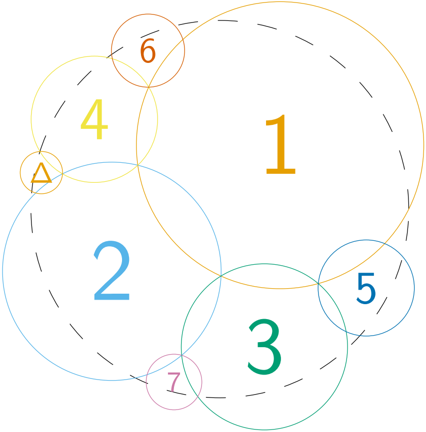
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic.



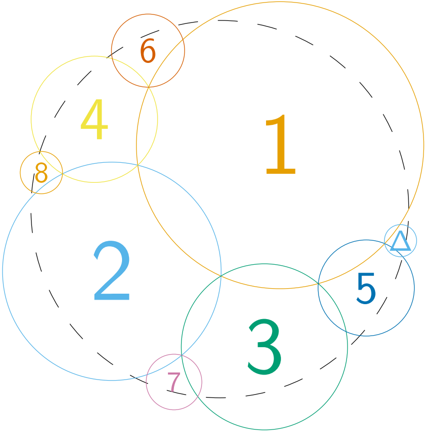
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic..



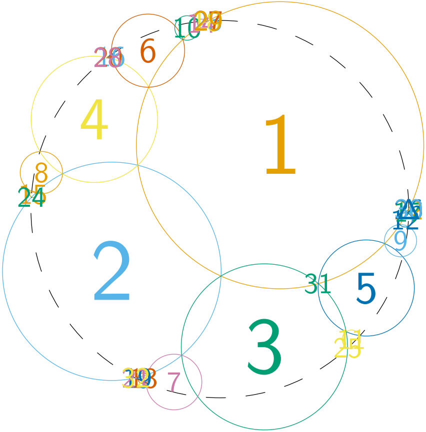
L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic...



L_2: Darting Non-Monotonic Algorithms – cont'd

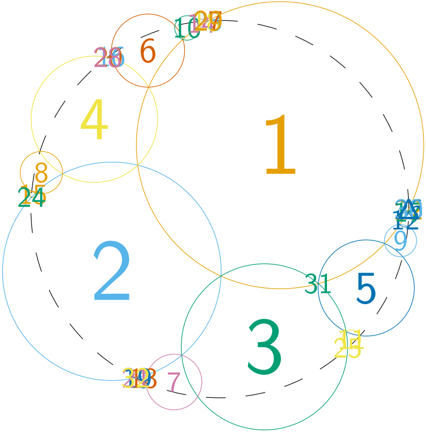
- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic............
- 32 probes!!!



L_2: Darting Non-Monotonic Algorithms – cont'd

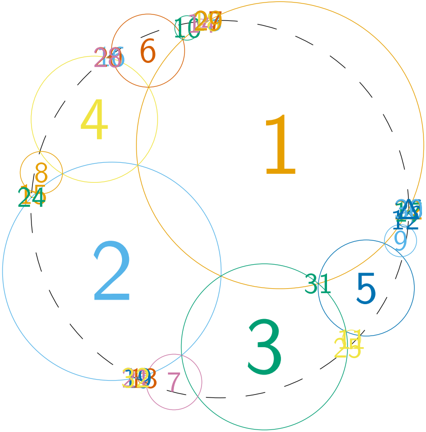
- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic............
- 32 probes!!!

- How good is it?



- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic............
- 32 probes!!!

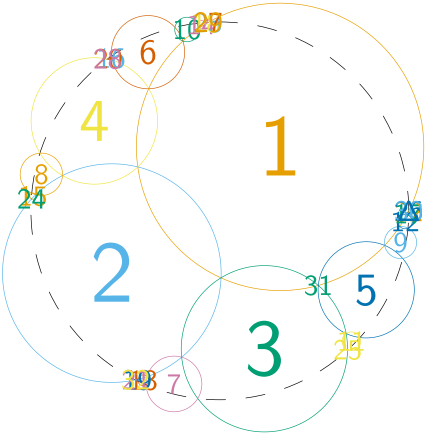
- How good is it?
- $P(n) \leq 2.53\lceil \log n \rceil$



L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic............
- 32 probes!!!

- How good is it?
- $P(n) \leq$ 2.53$\lceil \log n \rceil$
- Recall: 2.4$\lceil \log n \rceil$ – lower bound

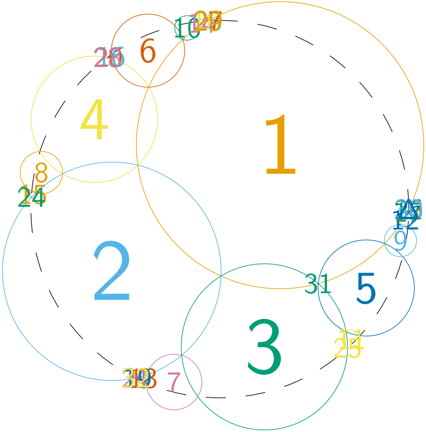


L_2: Darting Non-Monotonic Algorithms – cont'd

- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic............
- 32 probes!!!

- How good is it?
- $P(n) \leq {\color{red}2.53}\lceil \log n \rceil$
- Recall: ${\color{red}2.4}\lceil \log n \rceil$ – lower bound

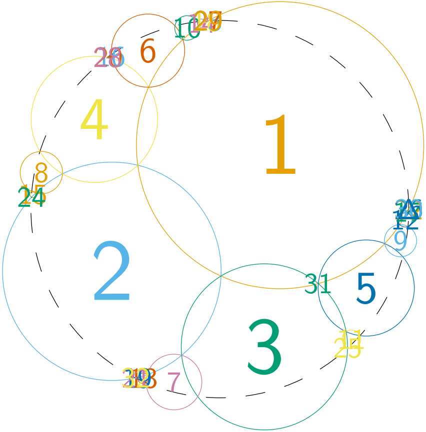
- Distance?



- How far can we push this?
- Take human out of the loop
- Differential evolution......
- Then programmatic............
- 32 probes!!!

- How good is it?
- $P(n) \leq 2.53\lceil \log n \rceil$
- Recall: $2.4\lceil \log n \rceil$ – lower bound

- Distance? abysmal
- $D(n) \leq 45.4n$



L_2: Comparing # Probes

Category	Alg. #	Probes ($P/\lceil\log n\rceil$) Min	Avg	Max	Bound
Hexagonal	Alg. 1	**1.00**	3.24	5.70	6.00
	Alg. 2	**1.00**	2.93	4.80	5.00
Chord-Based	Alg. 3	3.85	4.13	4.25	4.08
	Alg. 4	3.10	3.52	3.70	3.54
Monotonic	Alg. 5	3.55	3.87	4.15	3.83
	Alg. 6	3.25	3.41	3.85	3.34
Darting	Alg. 7	2.90	2.99	3.65	2.93
	Alg. 8	2.55	**2.59**	**3.20**	**2.53**

Table 1: A numerical comparison of simulation results for our 8 algorithms on the number of probes made (P). The best values are highlighted in bold.

L_2: Comparing Distance Traveled

Category	Alg. #	Total Distance (D/n) Min	Avg	Max	Bound
Hexagonal	Alg. 1	**0.00**	3.35	10.39	10.39
	Alg. 2	**0.00**	2.65	8.81	8.81
Chord-Based	Alg. 3	4.69	5.46	6.56	6.95
	Alg. 4	4.30	5.38	9.00	9.31
Monotonic	Alg. 5	**0.00**	**1.92**	6.72	6.72
	Alg. 6	**0.00**	1.96	**6.01**	**6.02**
Darting	Alg. 7	3.86	5.97	25.74	25.80
	Alg. 8	2.44	4.05	42.58	45.40

Table 2: A numerical comparison of simulation results for our 8 algorithms on the total distance traveled by Δ (D). The best values are highlighted in bold.

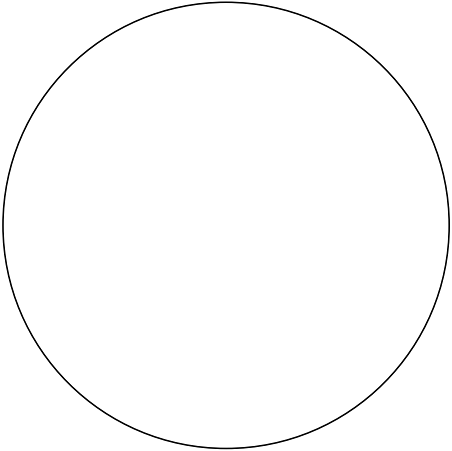
L_2: Comparing # Responses

Category	Alg. #	Responses ($R/\lceil \log n \rceil$)			
		Min	Avg	Max	Bound
Hexagonal	Alg. 1	**0.20**	**0.89**	**1.00**	**1.00**
	Alg. 2	0.35	1.11	1.45	2.00
Chord-Based	Alg. 3	1.40	1.99	2.40	4.08
	Alg. 4	0.80	1.94	2.50	3.54
Monotonic	Alg. 5	0.80	2.49	3.85	3.83
	Alg. 6	0.60	1.96	3.35	3.34
Darting	Alg. 7	0.30	1.39	2.15	2.93
	Alg. 8	0.45	1.31	1.85	2.53

Table 3: A numerical comparison of simulation results for our 8 algorithms on the number of POI responses made (R). The best values are highlighted in bold.

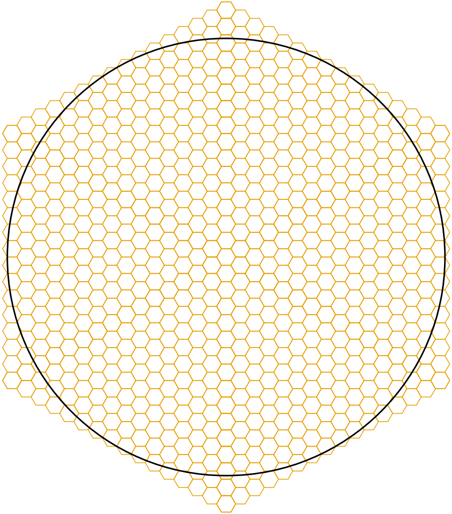
L_2: Reducing Responses

- If POI only allowed 1 response?



L_2: Reducing Responses

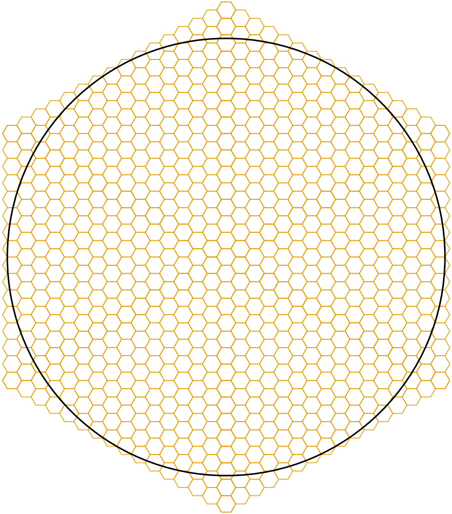
- If POI only allowed 1 response?
- Large hexagonal lattice!



L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

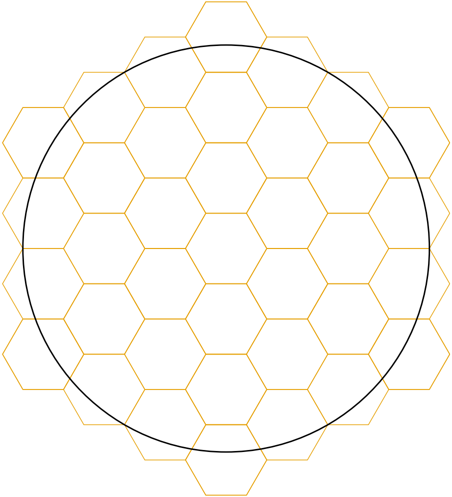
- If POI allowed 2 responses?



L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

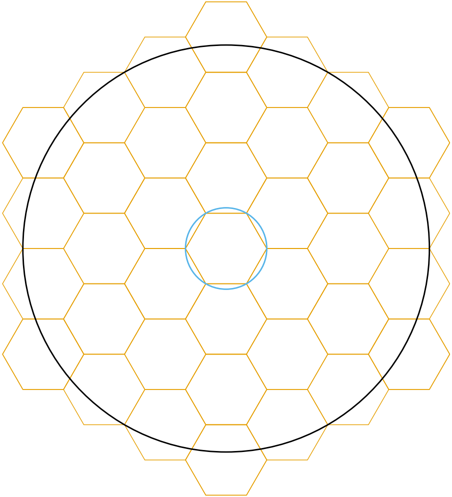
- If POI allowed 2 responses?
- Medium hexagonal lattice...



L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

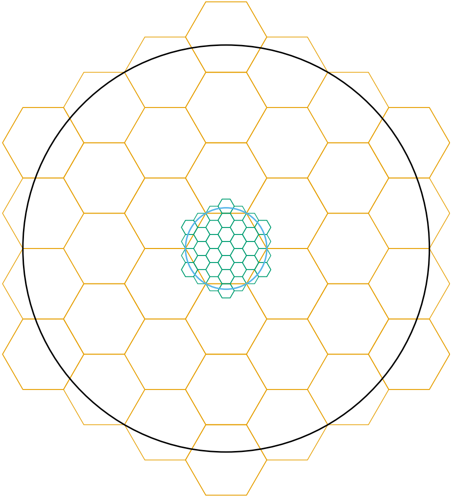
- If POI allowed 2 responses?
- Medium hexagonal lattice...
- After one response...



L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

- If POI allowed 2 responses?
- Medium hexagonal lattice...
- After one response... recurse!

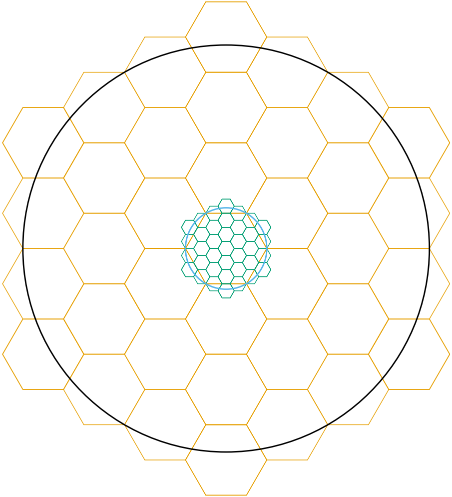


L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

- If POI allowed 2 responses?
- Medium hexagonal lattice...
- After one response... recurse!

- If POI allowed $R_{\max}$ responses?



L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

- If POI allowed 2 responses?
- Medium hexagonal lattice...
- After one response... recurse!

- If POI allowed R_{max} responses?
- R_{max} recursions!

Theorem

If a POI is allowed R_{max} responses,

$$P(n) \leq 6R_{max}\binom{\left\lceil \frac{2n^{\frac{1}{R_{max}}}+2}{3} \right\rceil}{2} \tag{1}$$

$$L = \left\lceil \frac{2n^{\frac{1}{R_{max}}}+2}{3} \right\rceil \text{ rings.} \tag{2}$$

L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

- If POI allowed 2 responses?
- Medium hexagonal lattice...
- After one response... recurse!

- If POI allowed $R_{\max}$ responses?
- $R_{\max}$ recursions!

Theorem

If a POI is allowed R_{max} responses,

$$P(n) \leq 6R_{max}\binom{\lceil \frac{2n^{\frac{1}{R_{max}}}+2}{3} \rceil}{2} \quad (1)$$

$$L = \lceil \frac{2n^{\frac{1}{R_{max}}}+2}{3} \rceil \text{ rings.} \quad (2)$$

Corollary

1. *If $R_{max} = 1$, $P(n) \leq \mathcal{O}(n^2)$.*
2. *If $R_{max} = 2$, $P(n) \leq \mathcal{O}(n)$.*
3. *If $R_{max} = \lceil \log n \rceil$, then $P(n) \leq 6\lceil \log n \rceil$.*

L_2: Reducing Responses

- If POI only allowed 1 response?
- Large hexagonal lattice!

- If POI allowed 2 responses?
- Medium hexagonal lattice...
- After one response... recurse!

- If POI allowed $R_{\max}$ responses?
- $R_{\max}$ recursions!

- Corollary 3 is Algorithm 1!

Theorem

If a POI is allowed R_{max} responses,

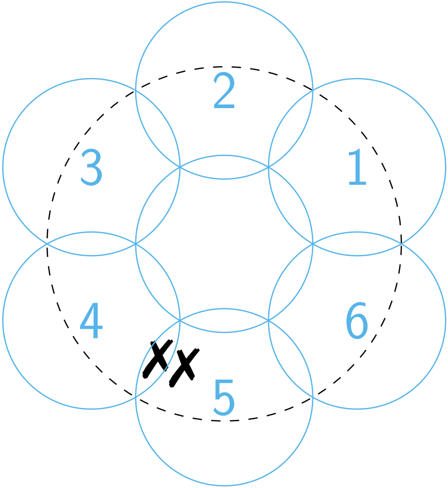
$$P(n) \leq 6R_{max}\binom{\lceil \frac{2n^{\frac{1}{R_{max}}}+2}{3} \rceil}{2} \tag{1}$$

$$L = \lceil \frac{2n^{\frac{1}{R_{max}}}+2}{3} \rceil \text{ rings.} \tag{2}$$

Corollary

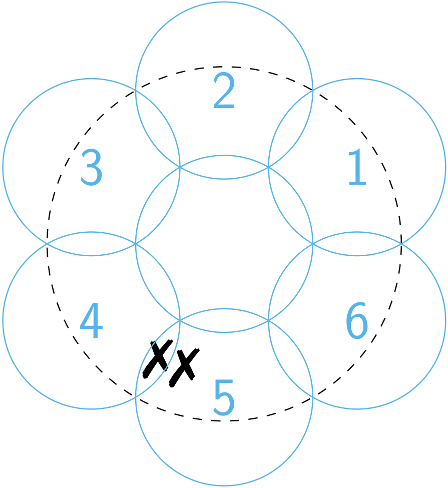
1. *If $R_{max} = 1$, $P(n) \leq \mathcal{O}(n^2)$.*
2. *If $R_{max} = 2$, $P(n) \leq \mathcal{O}(n)$.*
3. *If $R_{max} = \lceil \log n \rceil$, then $P(n) \leq 6\lceil \log n \rceil$.*

- Finding all k POIs?



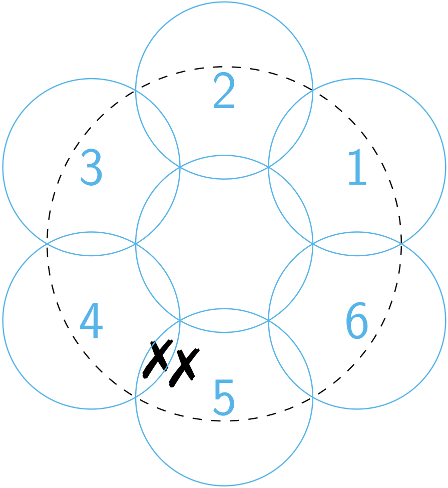
L_2: Finding All POIs

- Finding all k POIs?
- Using algorithm $\mathcal{A}(n)$



L_2: Finding All POIs

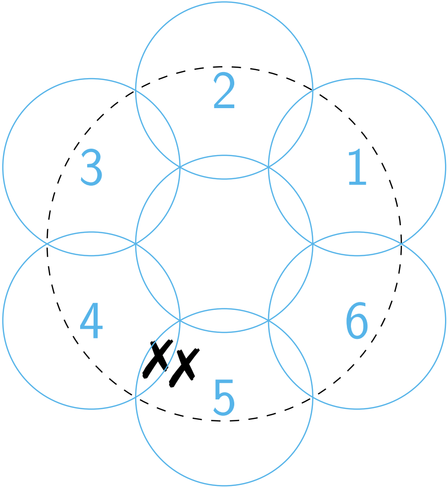
- Finding all k POIs?
- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq {\color{red}c}\lceil \log n \rceil$



L_2: Finding All POIs

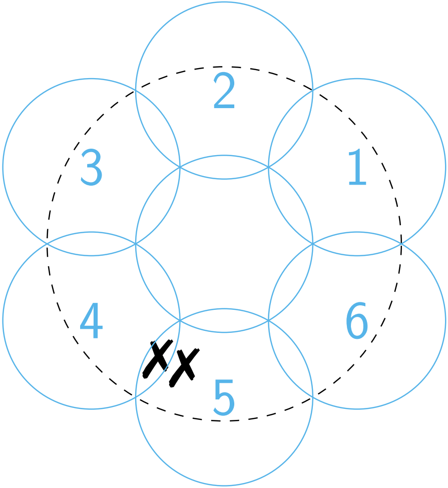
- Finding all k POIs?

- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq c\lceil \log n \rceil$
- Traveling $D(n) \leq dn$



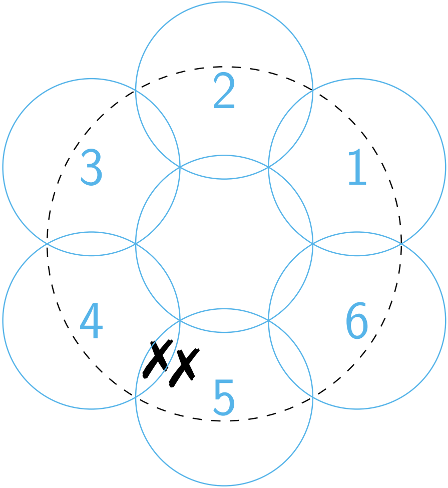
L_2: Finding All POIs

- Finding all k POIs?
- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq c\lceil \log n \rceil$
- Traveling $D(n) \leq dn$
- Trivial: Call $\mathcal{A}(n)$ k times



L_2: Finding All POIs

- Finding all k POIs?
- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq c\lceil \log n \rceil$
- Traveling $D(n) \leq dn$
- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil \log n \rceil$
- $D_{\text{tot}} \leq dkn$



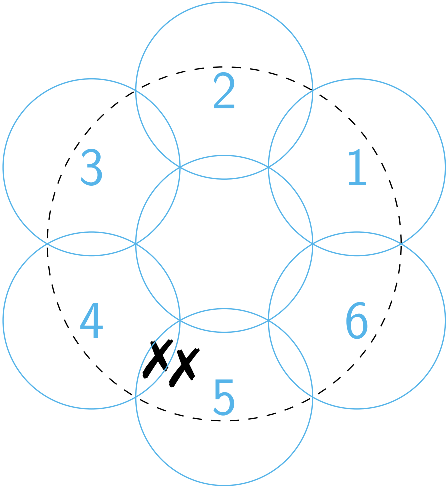
L_2: Finding All POIs

- Finding all k POIs?

- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq c\lceil \log n \rceil$
- Traveling $D(n) \leq dn$

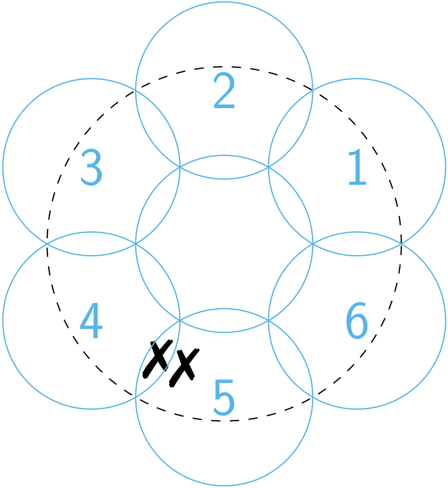
- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil \log n \rceil$
- $D_{\text{tot}} \leq dkn$

- Can we do better?



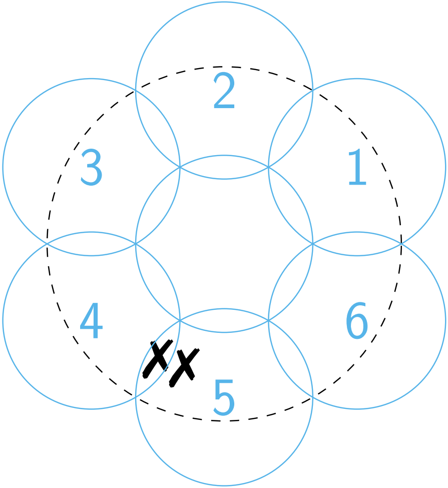
L_2: Finding All POIs

- Finding all k POIs?
- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq c\lceil \log n \rceil$
- Traveling $D(n) \leq dn$
- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil \log n \rceil$
- $D_{\text{tot}} \leq dkn$
- Can we do better?
- Recall: Probes return boolean



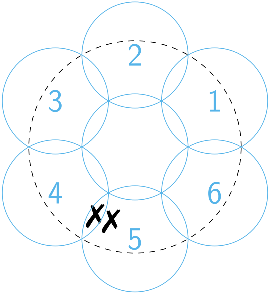
L_2: Finding All POIs

- Finding all k POIs?
- Using algorithm $\mathcal{A}(n)$
- One POI: $P(n) \leq c\lceil\log n\rceil$
- Traveling $D(n) \leq dn$
- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil\log n\rceil$
- $D_{\text{tot}} \leq dkn$
- Can we do better?
- Recall: Probes return boolean
- Even returning quantity...?



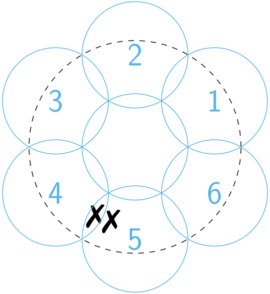
L_2: Finding All POIs – cont'd

- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil \log n \rceil$
- $D_{\text{tot}} \leq dkn$



- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil\log n\rceil$
- $D_{\text{tot}} \leq dkn$

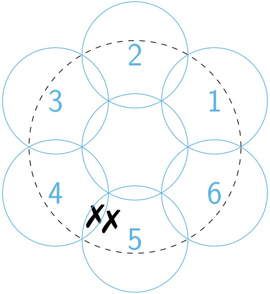
- Simple idea – once one found,



L_2: Finding All POIs – cont'd

- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \le ck\lceil \log n \rceil$
- $D_{\text{tot}} \le dkn$

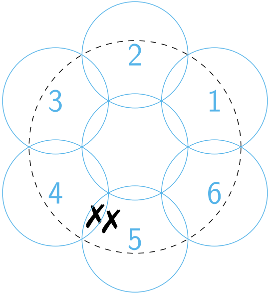
- Simple idea – once one found,
- exponential search



L_2: Finding All POIs – cont'd

- Trivial: Call $\mathcal{A}(n)$ k times
- $P_{\text{tot}} \leq ck\lceil \log n \rceil$
- $D_{\text{tot}} \leq dkn$

- Simple idea – once one found,
- exponential search



Theorem

We can perform a memoryless search for all k POIs in

$$P_{tot} \leq c\lceil \log n \rceil + (c+1)(k-1)\lceil \log \overline{e} \rceil,$$
$$D_{tot} \leq dn + 2dE,$$

where $E < OPT(\lceil \log k \rceil + 1)$, $\overline{e} = \frac{E}{k-1}$, *and OPT is the optimal tour length for the traveling salesperson problem (TSP) on the k POIs.*

Open Problems

- $P(n)$: Progressive probe LB: 2.40001, Alg. 6 achieves 2.53
  Can we tighten?
- Take advantage of known empty regions?
- Alg. 6 $D(n) \leq 6.02n$ – can we improve?
- Higher dimensions?
- Better find-all strategy?
- Other distance metrics (L_1, L_∞)?
- Instance optimal w.r.t. δ_{min}?