Fundamental Research

Scientific Perspectives on GMT 2019/2020

Marc van Kreveld

Fundamental Research

- Fully defined problem statement
- In a completely specified framework/setting
- Search for universal truths (no data)
- For GMT: with relevance to games, virtual environments, interaction, visualization, multimedia, …

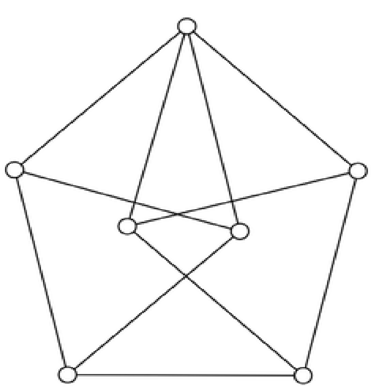
Fundamental research

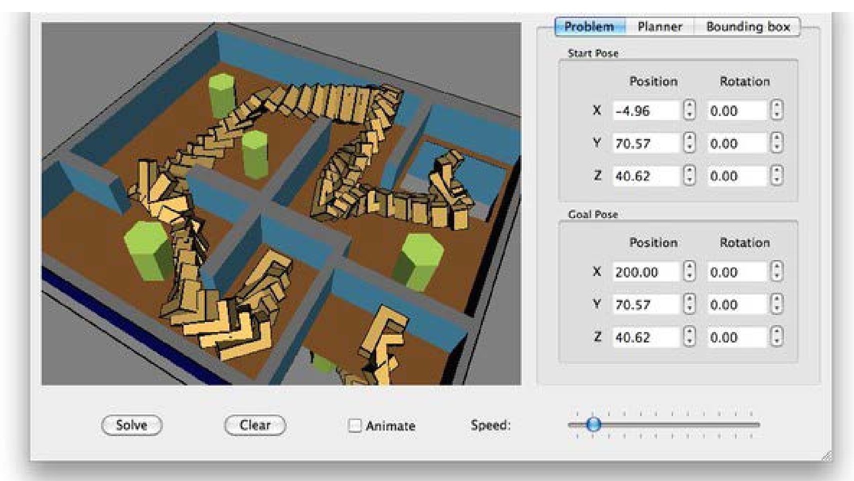
- Research for which data or people are not needed

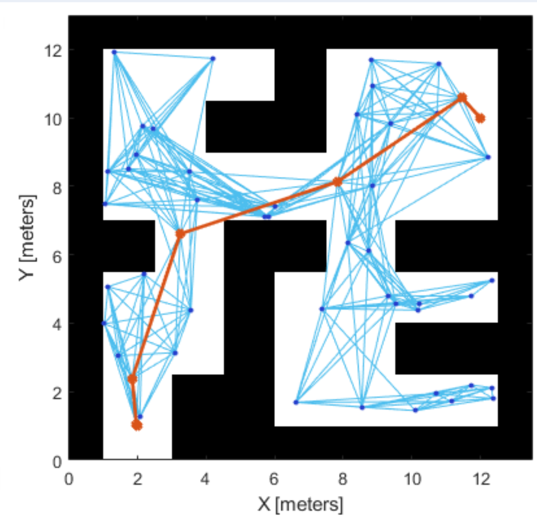
- Synonyms: foundational research, pure research
- Complementary to experimental research, user studies, …

Fundamental research

	Fundamental research question	Applied research question
Fundamental research approach	Can any planar graph with *n* vertices be drawn planarly with all vertices on an *n* x *n* integer grid and straight edges?	Can we compute a motion for a robot amidst polygonal obstacles, if one exists?
Experimental research approach	Can all randomly generated planar graphs with *n* vertices be drawn planarly on an *n* x *n* grid and straight edges?	Does the probabilistic path planner always find a motion for a robot, if one exists?







Types of fundamental research

- Algorithmic efficiency
- Approximation factor
- Competitive ratio
- Probabilistic bounds
- Comparisons on models of computation
- Comparisons of measures or models

Methods of fundamental research

- Proofs
  - By induction
  - By contradiction
  - By construction
  - By algebra (formula manipulation)
- Algorithm design
- Constructions

Formulate results as lemmas or theorems

Interest of fundamental research

- Intrinsic: interesting question on its own, curiosity
- Link to (an abstraction of) an application

Algorithmic efficiency

- Efficiency of algorithms expressed using order notation in size of the input (and sometimes output)
- Complexity classes
  - PSPACE-hardness
  - NP-hardness
  - Polynomial-time solvable
- Efficiency proofs
  - Worst-case upper bounds – all inputs
  - Worst-case lower bounds – one input class

Approximation

- Constant-factor approximation
- PTAS: for any constant $\varepsilon > 0$, there is a polynomial-time $(1+\varepsilon)$-approximation algorithm (running time may depend exponentially on ε, like in $O(2^{\varepsilon} n^2)$)
- LTAS: polynomial → linear (dependency on n)
- FPTAS: running time depends polynomially on n and ε, like in $O(n^2/\varepsilon^3)$
- log n / root(n)-approximation (not constant because approximation quality depends on n, the input size)

Competitive ratio

- For strategies to deal with unknown information
- "the cost of not knowing": how much worse might we do compared to if we knew (in ratio)?

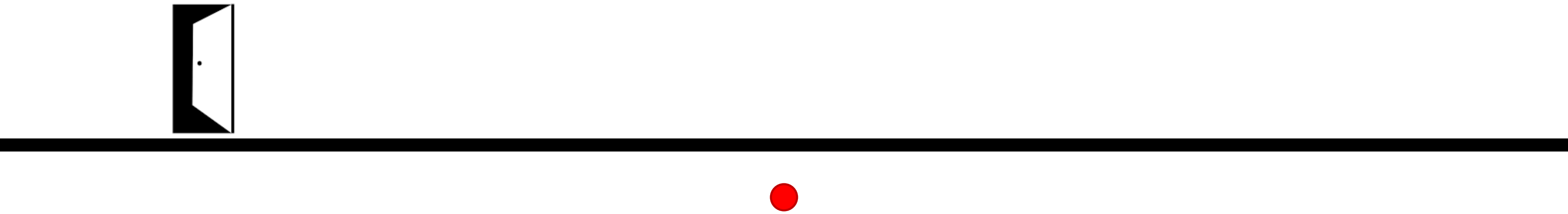
- Most commonly: search strategies where either the target or the environment (or both) is unknown

Searching: competitive analysis

- Best known: finding a door in a wall

Searching: competitive analysis

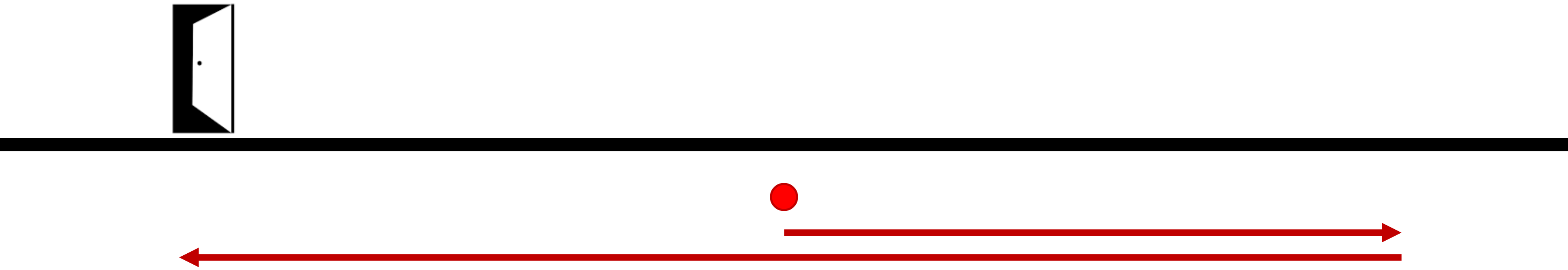
- Best known: finding a door in a wall



Assume we know the door is *D* away, but we do not know if it is left or right

Searching: competitive analysis

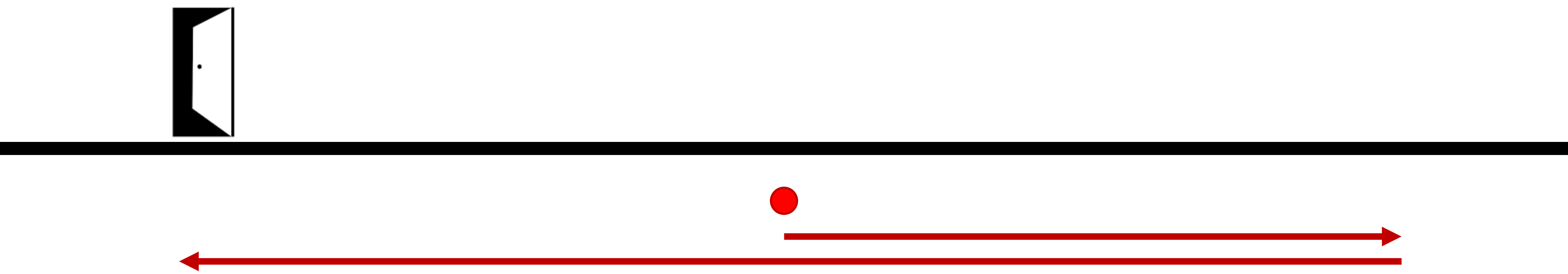
- Best known: finding a door in a wall



Assume we know the door is *D* away, but we do not know if it is left or right

Searching: competitive analysis

- Best known: finding a door in a wall



Assume we know the door is *D* away, but we do not know if it is left or right

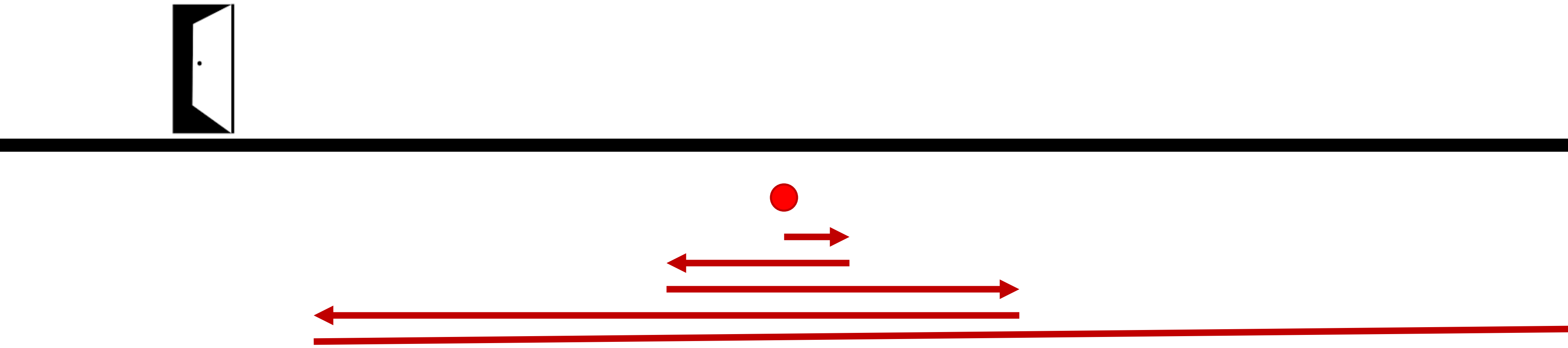
We walk *D* or 3*D*; she who knows, walks only *D*
→ 3-competitive strategy

Searching: competitive analysis

- **Theorem**: Given a searcher looking for a point on the real line who knows the exact distance to the point, there exists a search strategy that is 3-competitive
- The theorem statement assumes certain terms and the model to be known (competitive, precise measurement of walking distance)
- Proof by giving the strategy and analyzing it

Searching: competitive analysis

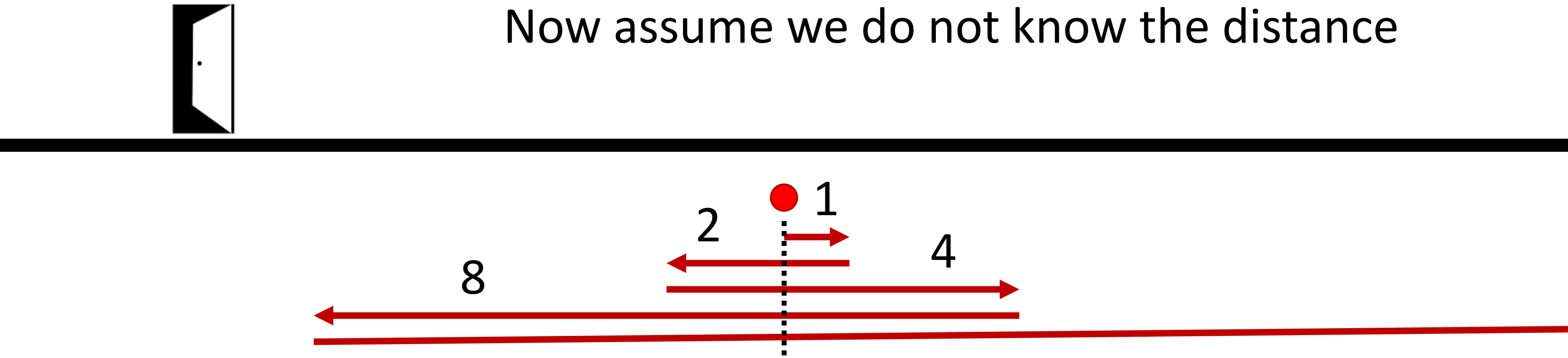
- Best known: finding a door in a wall



Searching: competitive analysis

- Best known: finding a door in a wall

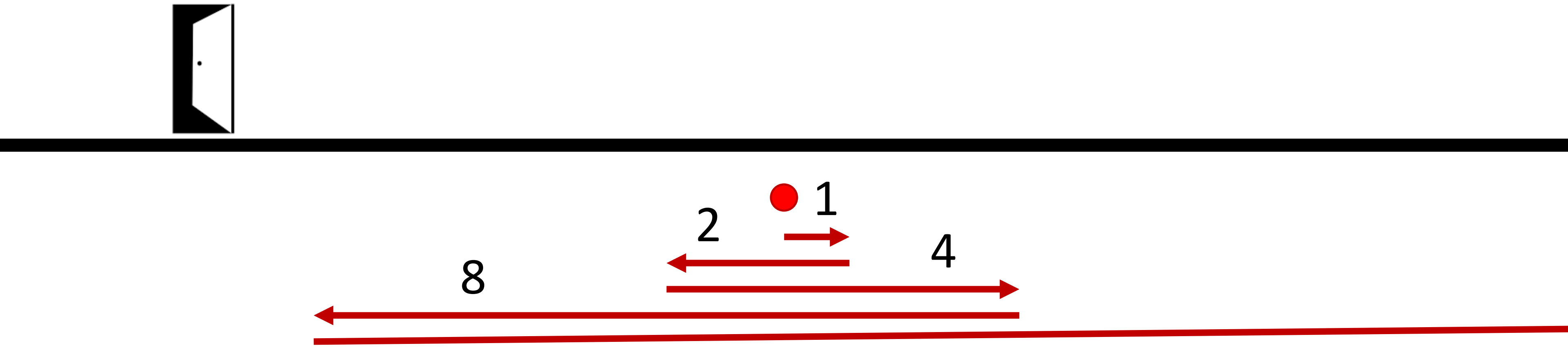
Now assume we do not know the distance



We walk: 1 + (1+2) + (2+4) + (4+8) + …

Searching: competitive analysis

- Best known: finding a door in a wall



Suppose the door is $D = 2^k + d$ away, $0 < d < 2^{k+2}$

We walk: $1 + (1+2) + \ldots + (2^k + 2^{k+1}) + 2^{k+1} + D =$

$2 \times 2^{k+2} - 2 + D < 8 \times 2^k + 2^k + d = 9 \times 2^k + d$

Searching: competitive analysis

- The competitive ratio is

$$\frac{\text{walked by strategy}}{\text{walked by knower}} = \frac{9 \times 2^k + d}{2^k + d} \quad \text{where } 0 < d < 2^{k+2}$$

This is maximized for *d* as small as possible (near zero), in which case the competitive ratio approaches 9; it is always < 9

Searching: competitive analysis

- **Theorem**: Given a searcher looking for a point on the real line, there exists a search strategy that is 9-competitive

- Not quite true ….

Searching: competitive analysis

- **Theorem**: Given a searcher looking for a point on the real line, there exists a search strategy that is 9-competitive, assuming the point is at least some known, arbitrarily small distance away from the starting position

- Proof by giving the strategy and analyzing it

Probabilistic bounds

- Random samples of a data set
- Random sampling of a real-world phenomenon
- Random choices in an algorithm

Models of computation

- What do we allow the machine model (Turing machine?) to do in a unit of time?
  - Any memory look-up or write
  - Any comparison or computation on two reals
  - Any root-finding of a constant-degree polynomial?
  - Trigonometric functions?
  - …
- What do we allow the machine model to do at all?

Models of computation

- What is harder to compute: the area or the perimeter of a simple polygon?

Models of computation

- What is harder to compute: the area or the perimeter of a simple polygon?

- Answer 1: Both can be done in O(n) time, if the polygon has n vertices

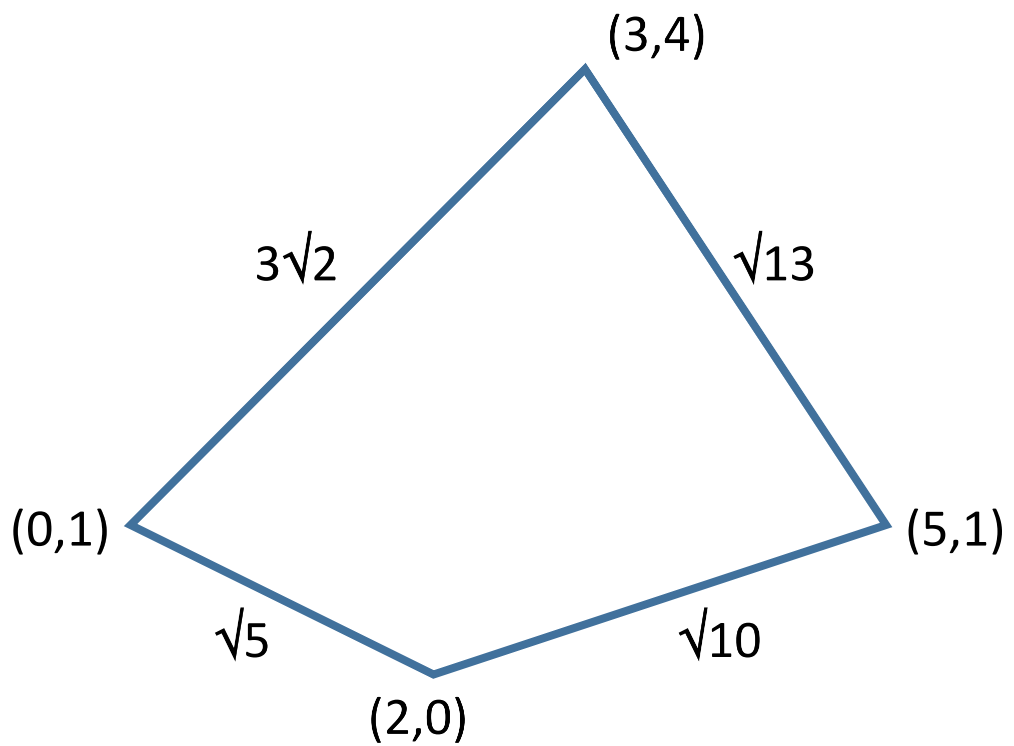
Models of computation

- What is harder to compute: the area or the perimeter of a simple polygon?
  Assume the vertices are given by integer coordinates

Models of computation

- What is harder to compute: the area or the perimeter of a simple polygon?
  Assume the vertices are given by integer coordinates

- Answer 2:
  - The area can be computed using only +, -, *, and divide by 2 on integers
  - The perimeter requires computing sums of square roots

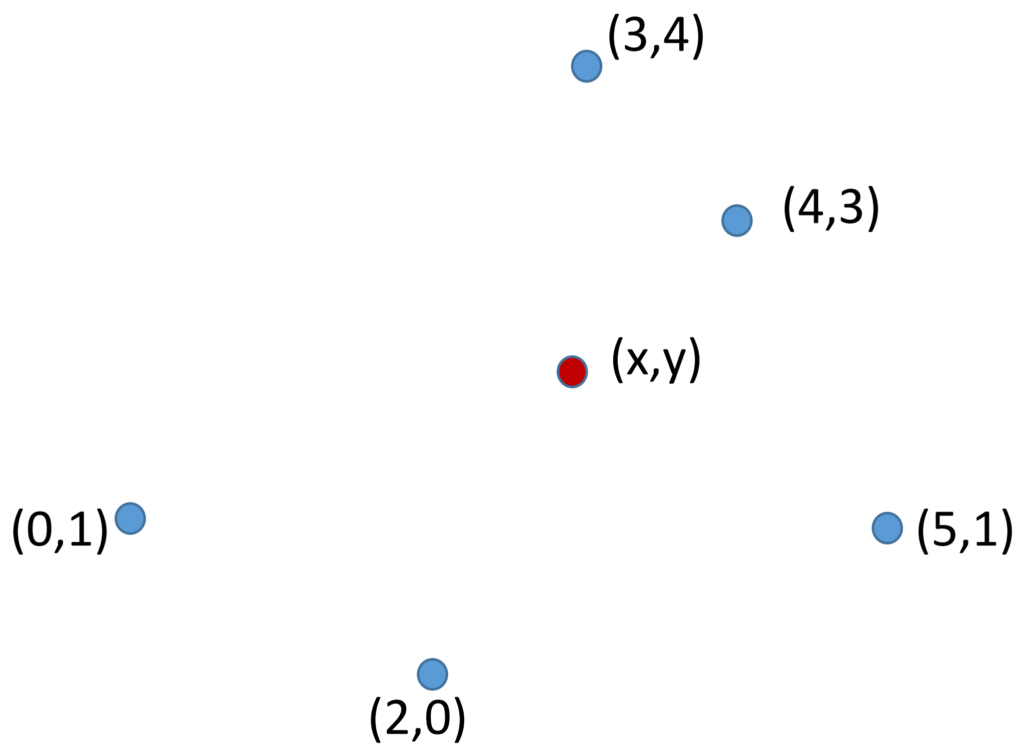


Perimeter = $3\sqrt{2} + \sqrt{13} + \sqrt{10} + \sqrt{5}$

Area = 10

Models of computation

- What is harder to compute: a point that minimizes the **sum of distances** to *n* other points, or a point that minimizes the **sum of squared-distances** to *n* other points?



Minimize:

$$x^2 + (y-1)^2 + (x-2)^2 + y^2 + (x-5)^2 + (y-1)^2 + (x-4)^2 + (y-3)^2 + (x-3)^2 + (y-4)^2$$

$\sqrt{}$ $\sqrt{}$ $\sqrt{}$ $\sqrt{}$ $\sqrt{}$

Models of computation

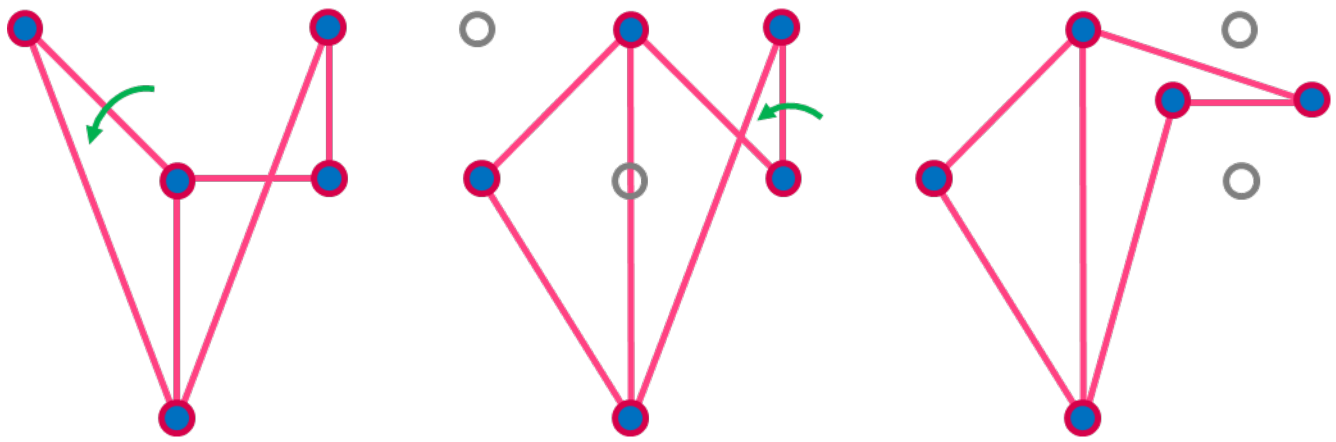
- Minimizing a quadratic function in *x* and *y* is easy, by computing partial derivatives. We need to find the root of a linear function
- Minimizing a function that has the unknowns in several different square roots is hard (even for a single unknown). No closed-form solution exists (a formula like the *abc*-formula or *pq*-formula for finding roots of a quadratic equation)

Models of computation

- What number type is needed to solve a problem?
  - Integers Z
  - Rational numbers (e.g., 2/3) Q
  - Algebraic numbers (e.g., root(2))
  - Transcendental numbers (e.g., π) R
  - Complex numbers (e.g., π + 3i) C

Puzzle game

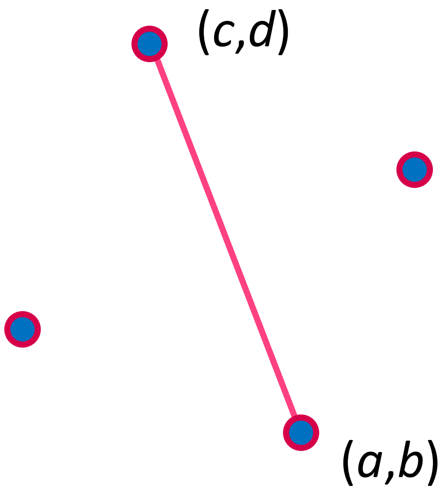
- Example from a puzzle game: 90 degree rotations



Recent master thesis project of Casper van Dommelen

Puzzle game

- In every step we may need extra precision worth one bit, but not more



New coordinates:

((*a*-*b*+*c*+*d*)/2, (*a*+*b*-*c*+*d*)/2)
((*a*+*b*+*c*-*d*)/2, (-*a*+*b*+*c*+*d*)/2)

If *a*, *b*, *c*, and *d* are integer, then the new coordinates may contain halves but not worse → one extra bit needed

(*very easy algebraic proof*)

Puzzle game

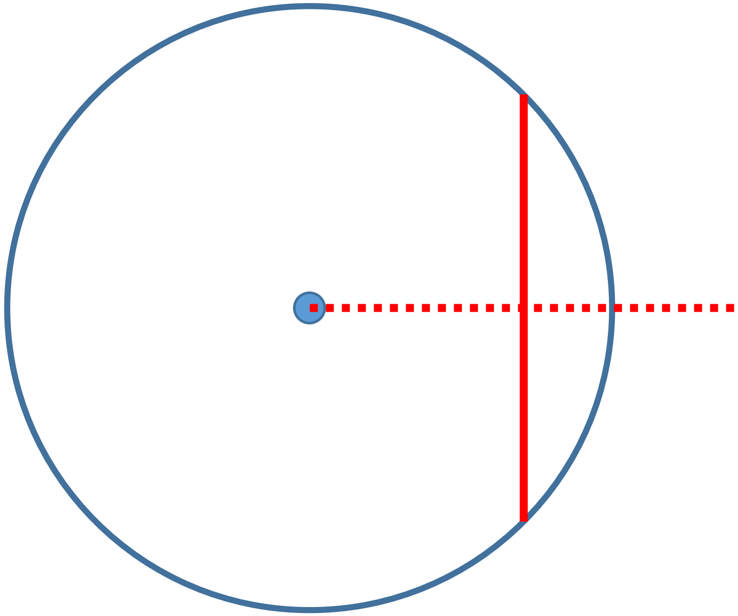
- **Theorem**: for any point set *P*, the center of mass of *P* does not change under a 90 degree rotate of two points of *P* about their midpoint
- Proof by algebra
- **Theorem**: for any point set *P*, the sum of squared distances to the center of mass does not change under a 90 degree rotate of two points of *P* about their midpoint
- Proof by algebra

Puzzle game

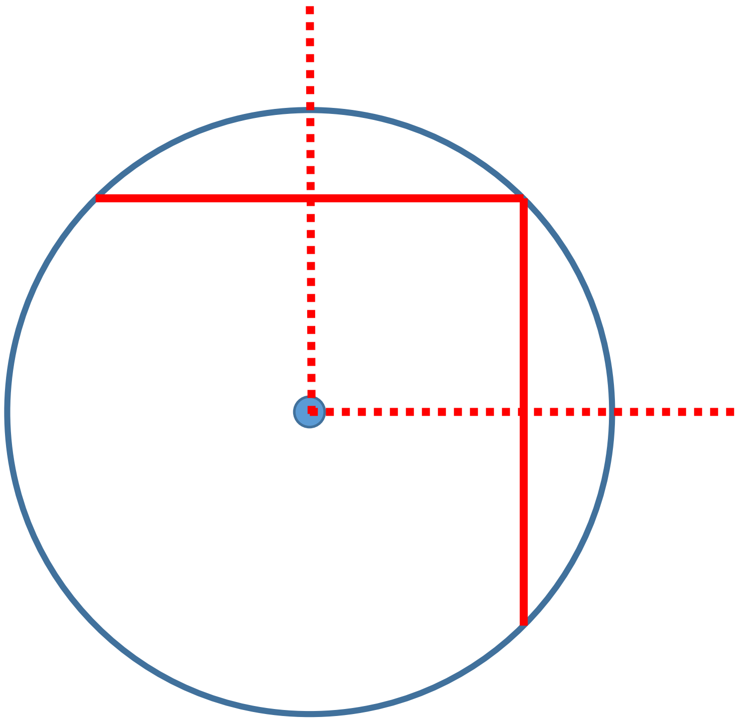
- Fundamental question: Is it possible to define a constant-size play area such that for any graph whose vertices lie in [0,1] x [0,1], any set of rotates stays within that play area?

- Possible answer YES: need proof
- Possible answer NO: need graph (construction) and rotate sequence that gets vertices arbitrarily far away (which is the basis of the proof)

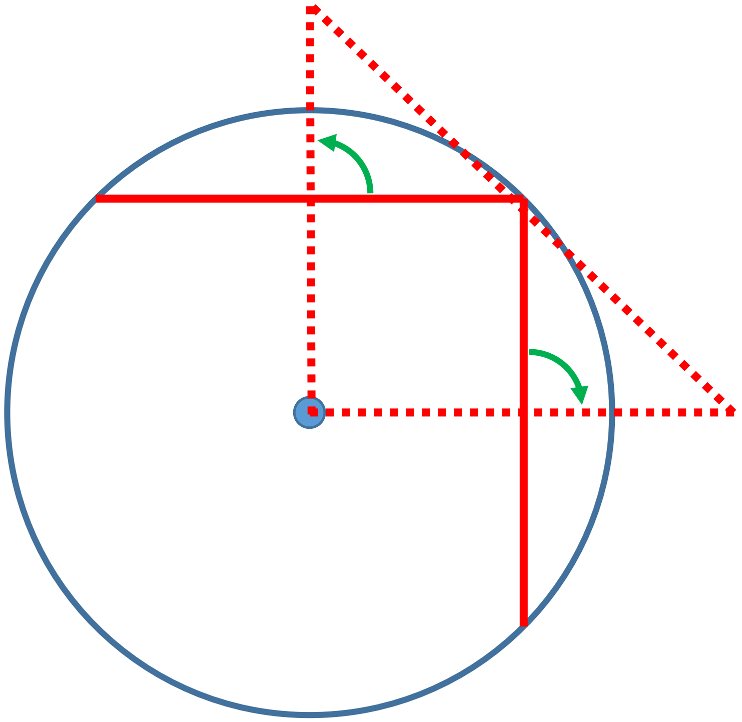
Puzzle game



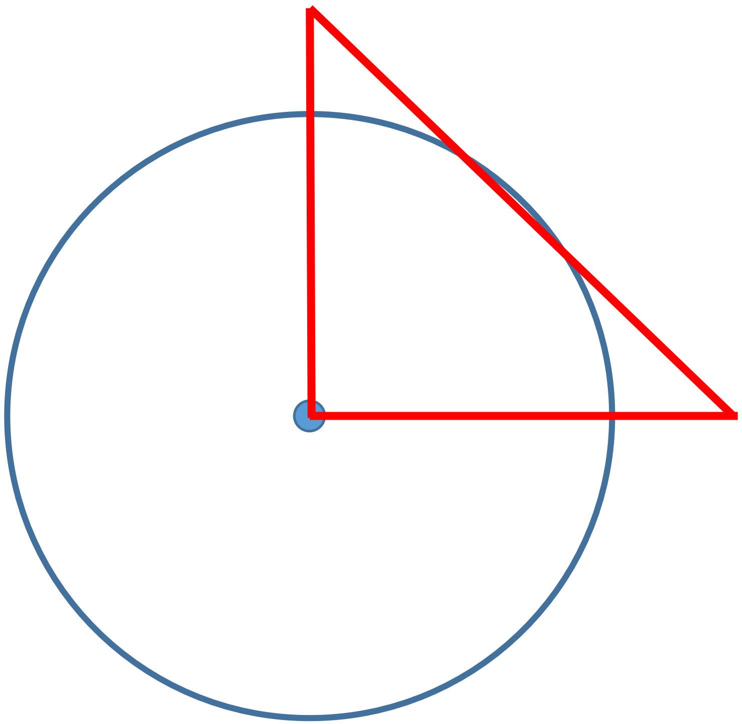
Puzzle game



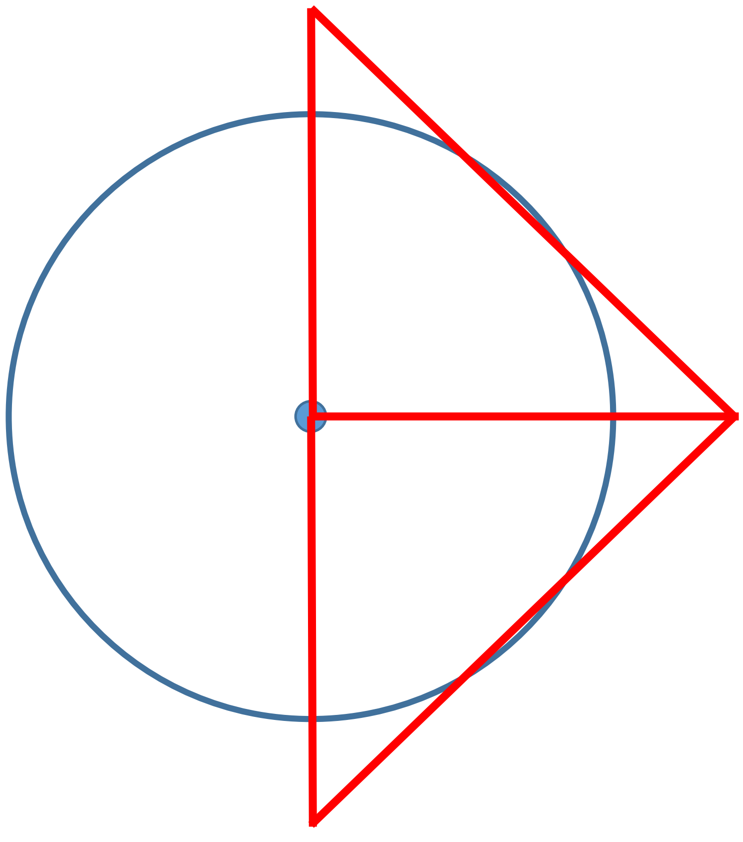
Puzzle game



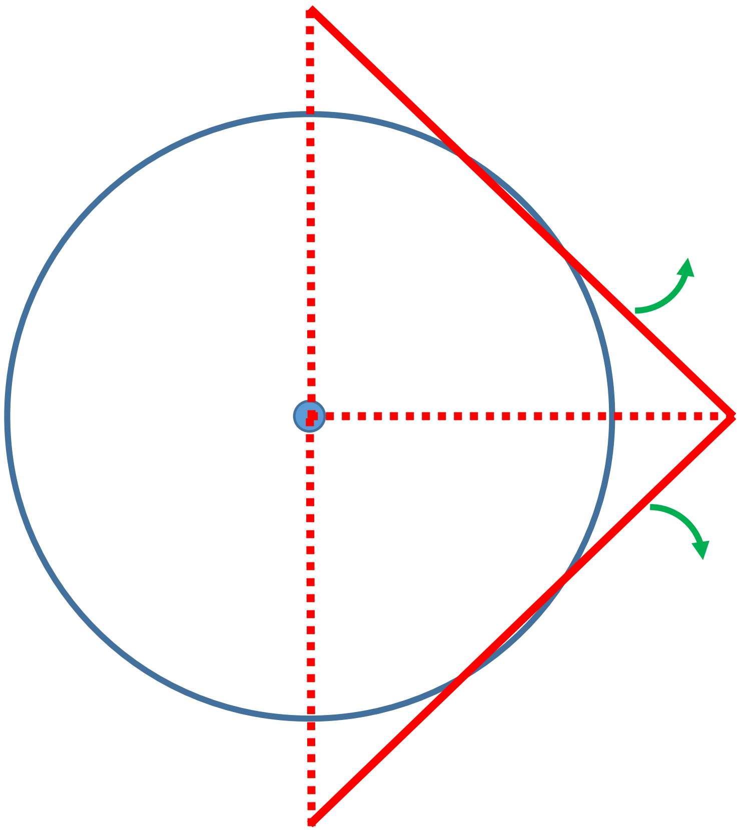
Puzzle game



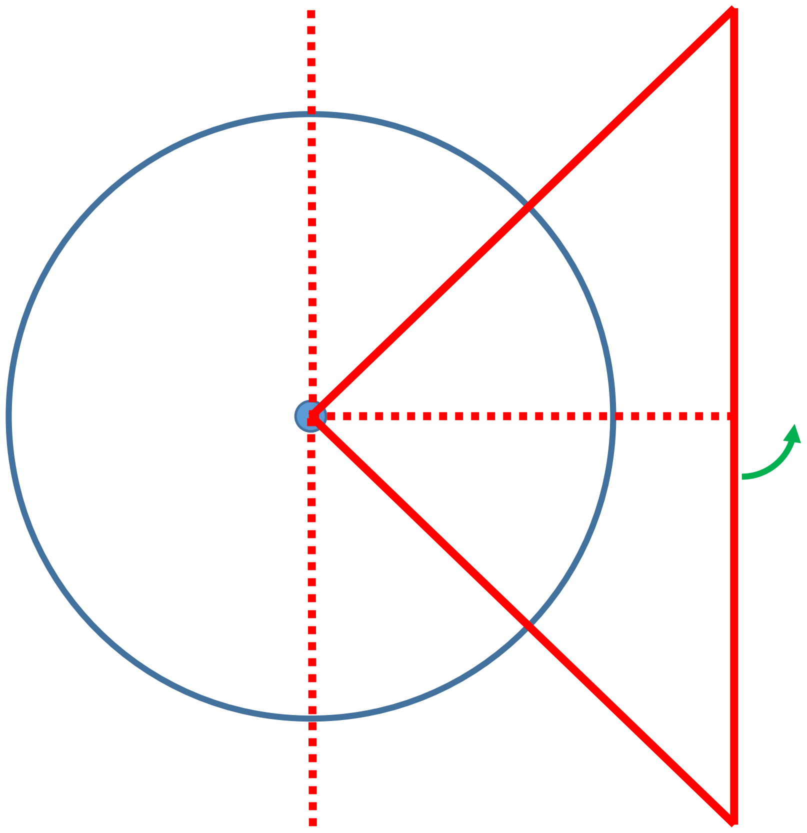
Puzzle game



Puzzle game



Puzzle game



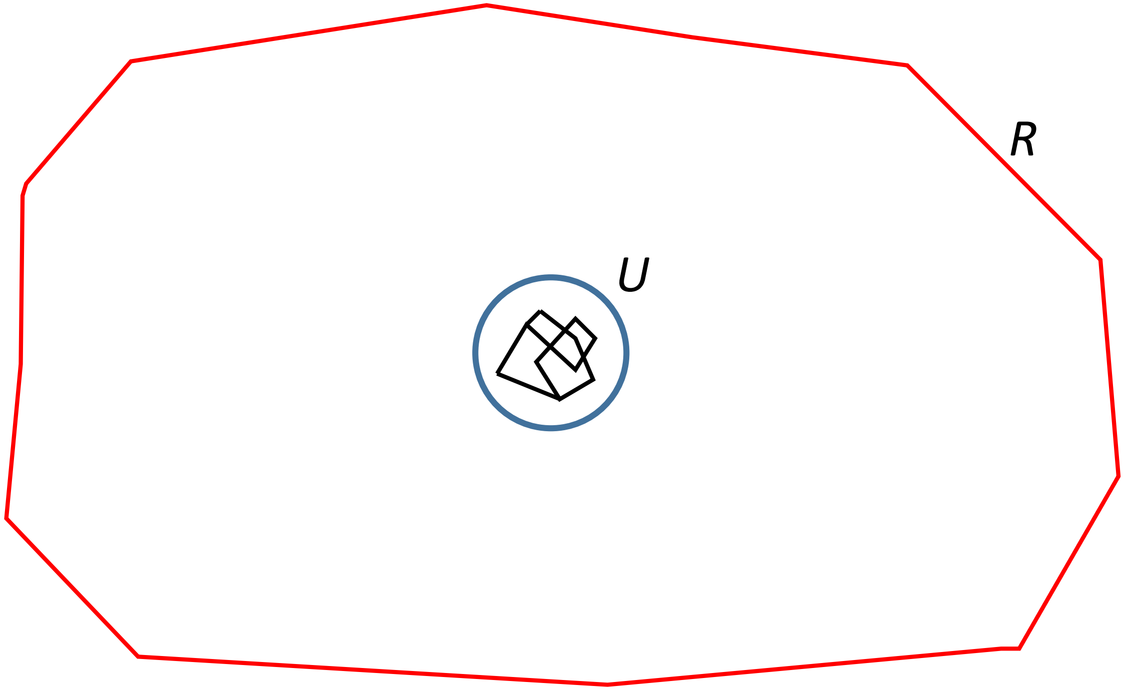
Puzzle game

- Fundamental question: Is it possible to define a constant-size play area such that for any graph whose vertices lie in [0,1] x [0,1], any set of rotates stays within that play area?

- Answer is NO: proof by construction: a graph class and rotate sequence that gets vertices arbitrarily far away

Puzzle game

- **Theorem:** For any bounded region *R* and any unit size region *U* inside it, a graph exists that starts in *U* and can get a vertex outside *R* after finitely many rotations



Puzzle game

- Any single graph cannot give the no-answer (it must be a graph *class*)

- Let *G* be any graph, assume it has *n* vertices, and assume they lie in a radius-1 disk
- Then the sum of squared distances is at most *n*
- So no vertex can be further than root(*n*) away from the center of mass

Puzzle game

- Any single graph cannot give the no-answer (it must be a graph *class*)
  - Let *G* be any graph, assume it has *n* vertices, and assume they lie in a radius-1 disk
  - Then the sum of squared distances is at most *n*
  - So no vertex can be further than root(*n*) away from the center of mass
- For any given graph, *n* is just a value, but a construction that works for *n* arbitrarily large is a graph class

Puzzle game

- The most interesting theoretical question (to me):

  Given any non-planar placement of a connected planar graph, can it always be made planar by rotate-90-degree moves?

- Why is “connected” in the statement?
- Do you see graph subclasses of planar graphs for which you can give an answer?

Puzzle game

- If no, what is the smallest counterexample?
- If yes, can the number of rotates be bounded by a function of *n*, the number of vertices in the graph?

A probabilistic bound

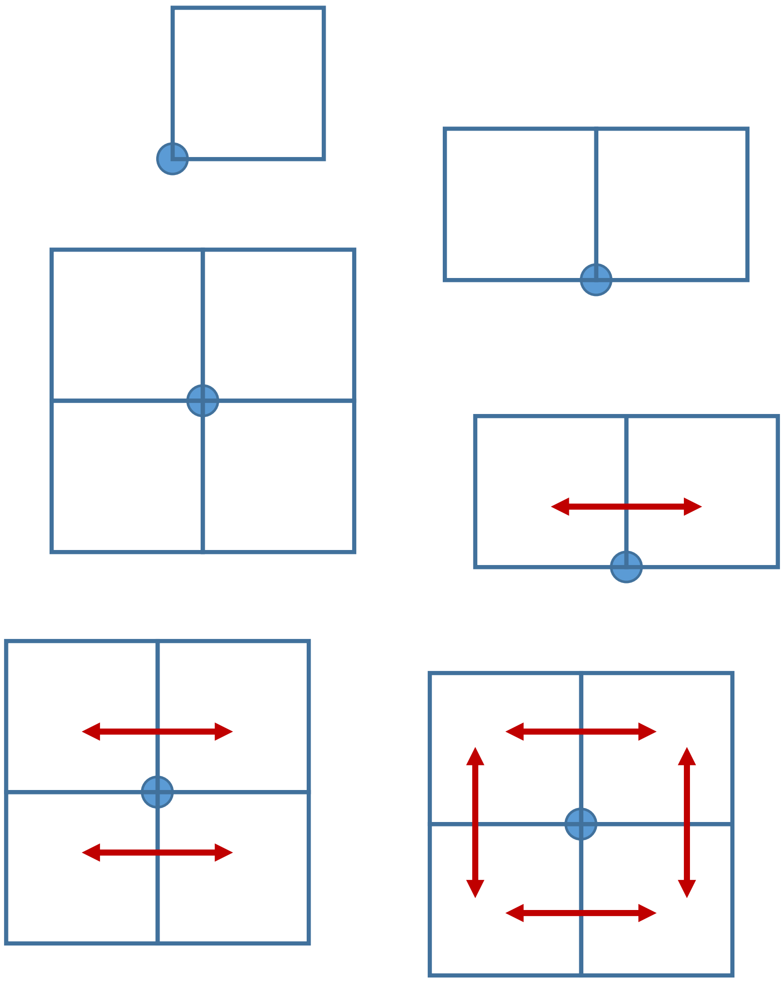
- Suppose *n* numbers are in random order in array A
- We find the max by going from A[1] to A[*n*] and updating *max* when we find a higher number
- What is the expected number of times we update *max*?

A probabilistic bound

- What is the probability that we update *max* after checking A[i]?
- This happens only if A[i] > A[1], …, A[i-1]
- Since A[1], …, A[i] are in random order the probability is 1/i
- The expected number of updates in total is $1/1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/n < \ln(n) + 1$

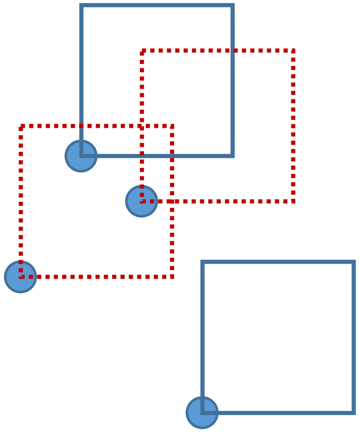
Model comparison

- Text placement on maps:
  - 1-position model
  - 2-position model
  - 4-position model
  - Any c-position model
  - 1-slider model
  - 2-slider model
  - 4-slider model
- All labels are assumed to be unit squares

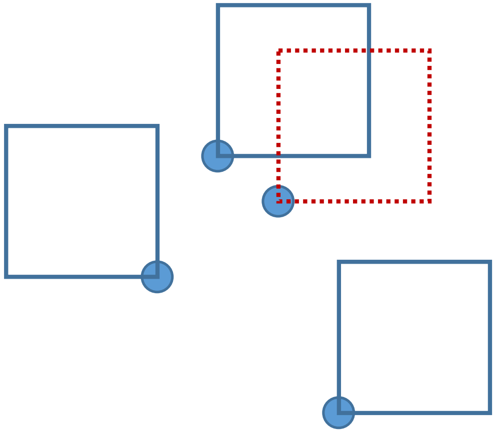


Model comparison

- How much better can one model be than another, in the worst (best?) case for any set of points to be labeled?



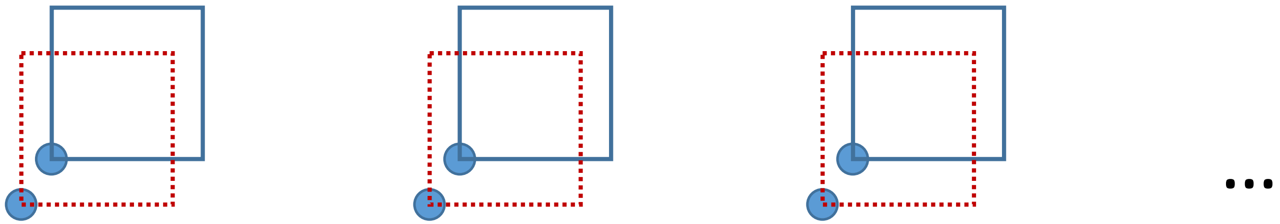
1-position model



2-position model

Model comparison

- The 2-position model can sometimes allow twice as many labels as the 1-position model (for arbitrarily large *n*): by construction of an example class



- Can it be even worse?

Model comparison

- The 2-position model never allows more than twice as many labels as the 1-position model:
  - Take an optimal solution in the 2-position model
  - If at least half the points have their label top-right, then we are done: the 1-position model can choose these
  - Otherwise, at least half the points have their label top-left
  - Choose these points but with a label top-right
  - These are non-intersecting if and only if the top-left ones were, because all labels move exactly one unit to the right

Model comparison

- **Theorem:** For any set of *n* points in the plane, if a disjoint labeling with unit squares exists of a subset of *k* points that are either to the top-left or top-right, then a disjoint labeling with unit squares exists of at least *k*/2 points that are to the top-right

  (For any *n*,) there exists a set of *n* points that allows *n* unit squares to the top-left or top-right, but only n/2 unit squares to the top-right

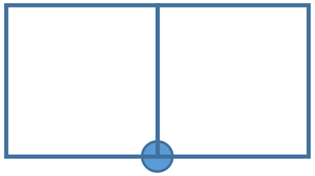
  (The top statement is a *worst-case optimal bound*: We have no hope of placing more than *k*/2 squares in all cases)

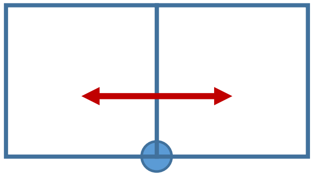
Model comparison

- Similar arguments can be used to compare 1-, 2- and 4-position models

Model comparison

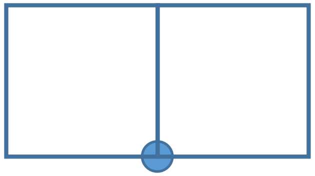
- How do the 2-position model and 1-slider model compare?

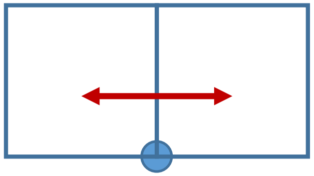




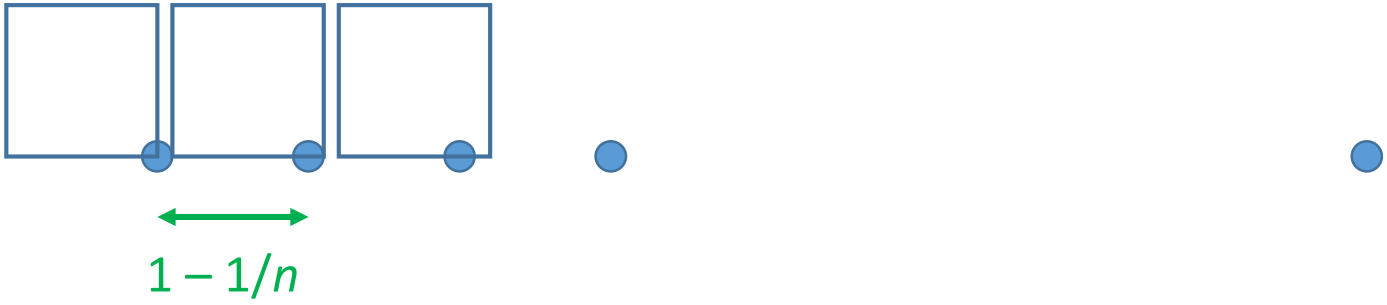
Model comparison

- How do the 2-position model and 1-slider model compare?





- Sometimes, the 1-slider model allows (nearly) twice as many labels, for any *n*



Model comparison

- This construction (class of points) allows all n points to be labeled in the 1-slider model and only $\lfloor n/2 \rfloor + 1$ points in the 2-position model

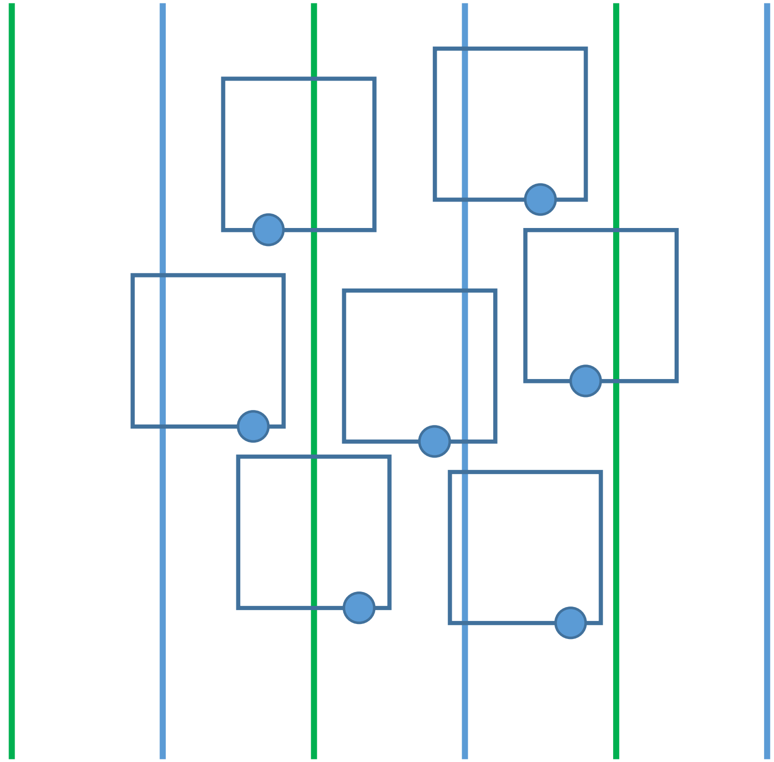
- Can it be worse?

Model comparison

- No, by proof
- Recall labels are unit squares
- Consider the lines x=0, x=1, x=2, x=3, …
- Consider an optimal solution in the 1-slider model: any placed label intersects exactly one line
- Take the labels intersecting the odd or even lines, whichever intersects more labels from the optimal 1-slider solution
  (by pigeonhole principle at least half)

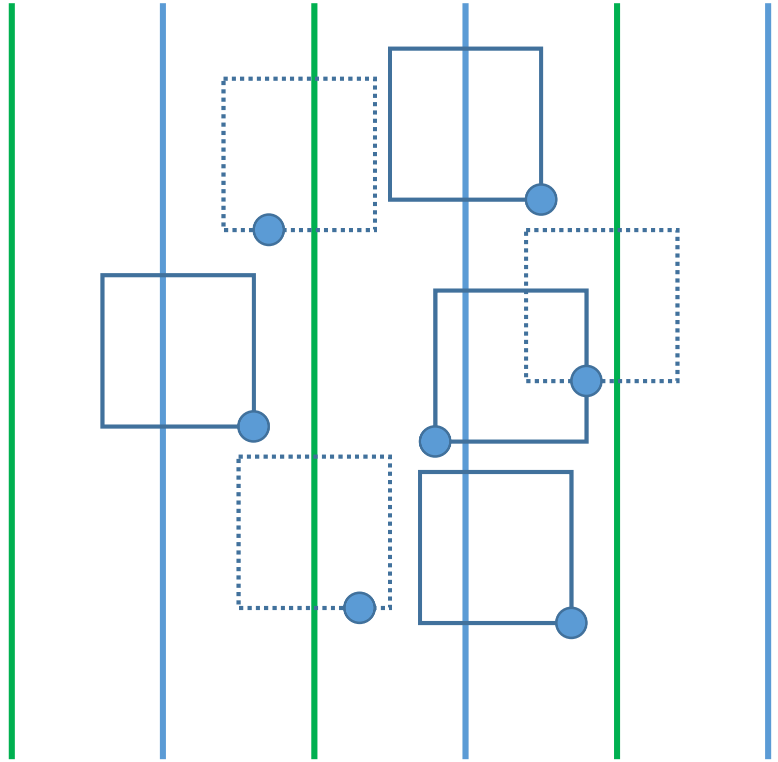
Model comparison

- Slide these labels left or right to a corner position of their square, while still intersecting the same line



Model comparison

- Slide these labels left or right to a corner position of their square, while still intersecting the same line



Squares intersecting different blue lines cannot intersect; squares intersecting the same blue line did not intersect in the 1-slider solution

Types of fundamental research

- Algorithmic efficiency – *well known*
- Approximation factor – *well known*
- Competitive ratio – *searching, unknown information*
- Probabilistic bounds – *sampling or randomized algorithms*
- Comparisons on models of computation – *complexity of the basic operations*
- Comparisons of measures or models

Results are given by lemmas and theorems