From Ants to Query Complexity

Amos Korman
IRIF, CNRS

Based on joint works with

Biologists: Ofer Feinerman, Ehud (Udi) Fonio, Yael Heyman, Aviram Gelblum
CS researchers: Lucas Boczkowski, Adrian Kosowski, Yoav Rodeh

Collective transport by Crazy ants

P. longicornis ants

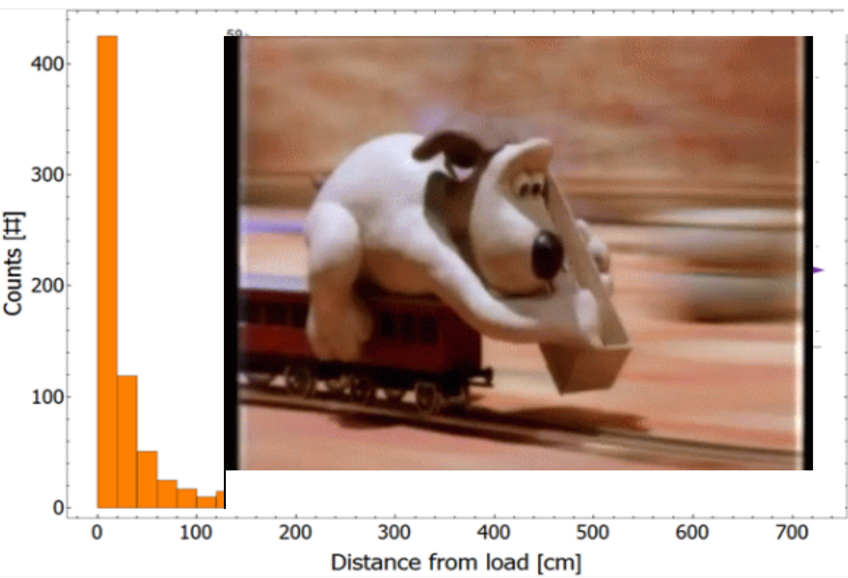
Method:
Scent mark detection

A new type of ant trail

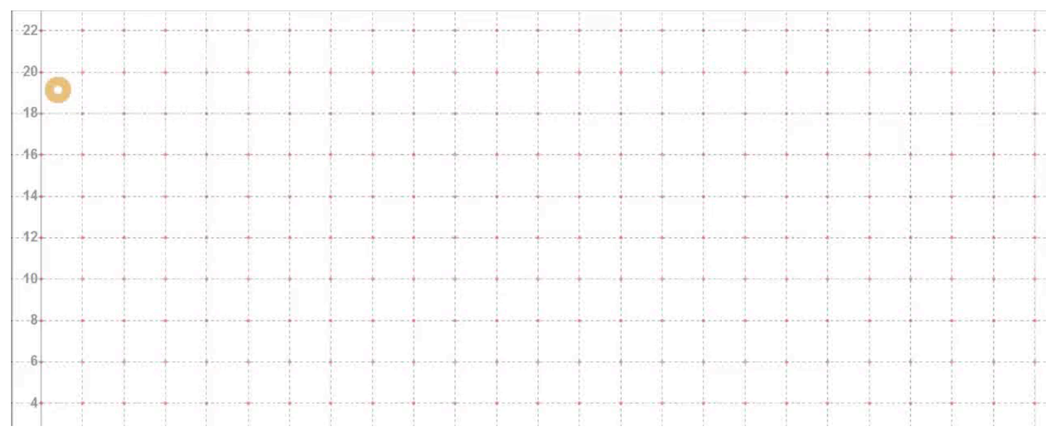
The known trail -
Very long and steady

100 meters

1. Individuals lay local trails



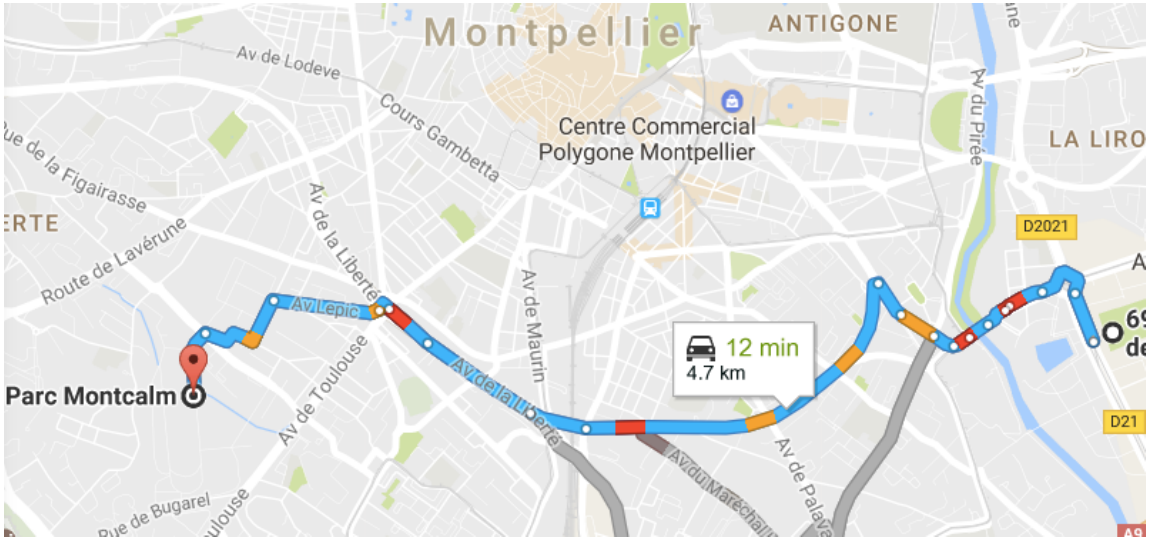
2. Group follows them, but not religiously



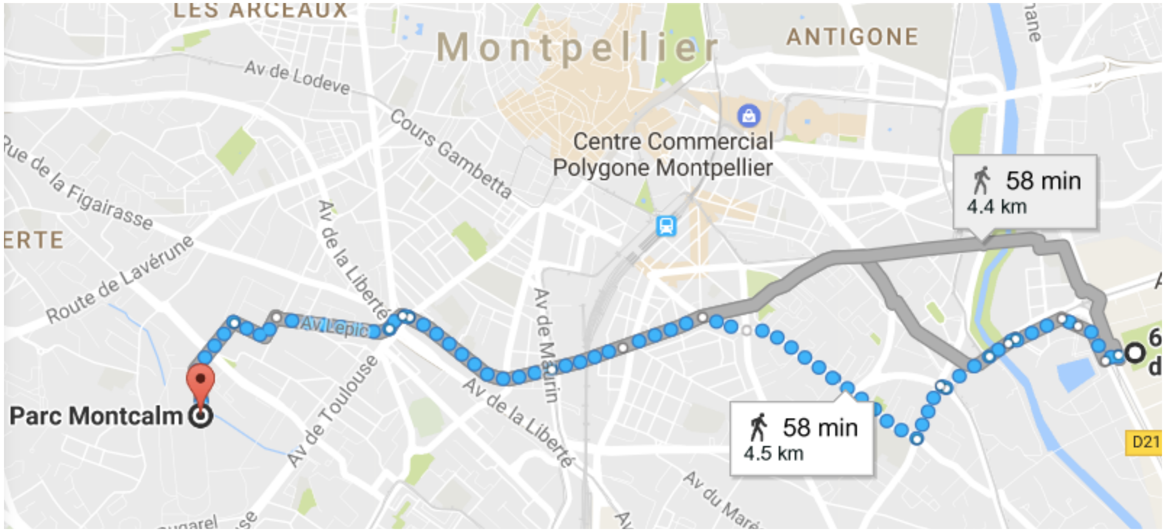
What is it good for?


To nest

You need to drive a car



You get the map for walking



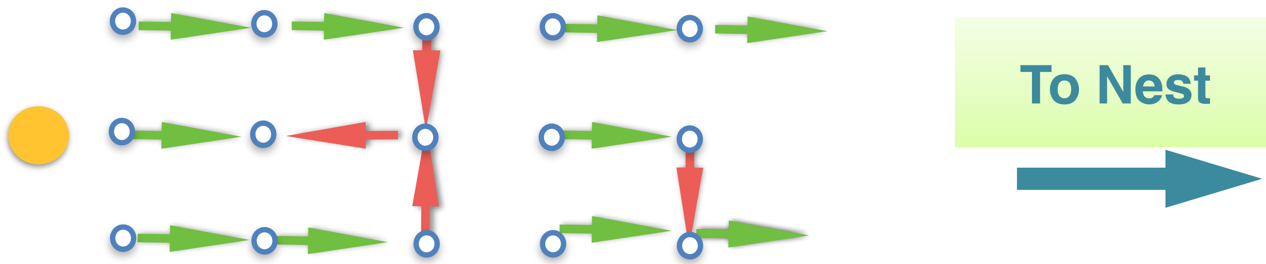
Navigating with unreliable roadsigns

Imagine driving in a foreign country after a hurricane
Most road signs are intact but some have been turned

How can you still get to your destination fast?

The Noisy Advice model

Each node has an advice pointer, which is **correct** with probability p.
Otherwise, w.p. q=1-p it points at an arbitrary direction



How can we reach the nest fast?

Advice per node is fixed!

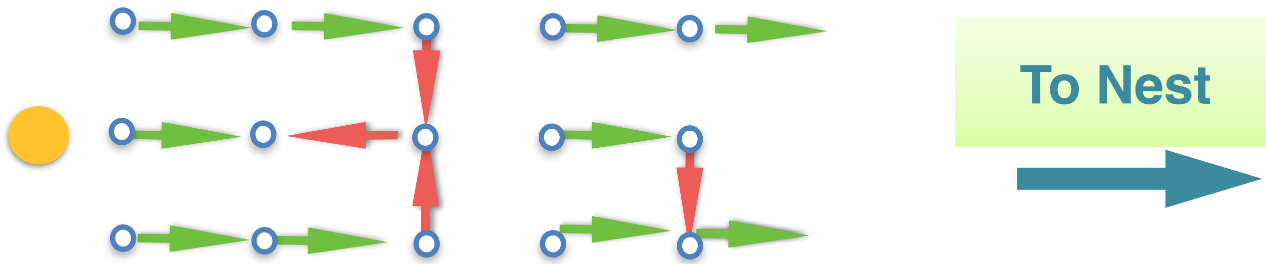
Very different from advice that can be resampled at any query (allows for busting)

[Feige et al. SICOMP'94] [Ben-Or and Hassidim FOCS'08]
[Emamjomeh-Zadeh et al. STOC'16] [Karp and Kleinberg SODA'07]

Very different from worst case placement of wrong advice

[Hanusse et al. PODC'10, TCS'08]

Random Listening (RL)

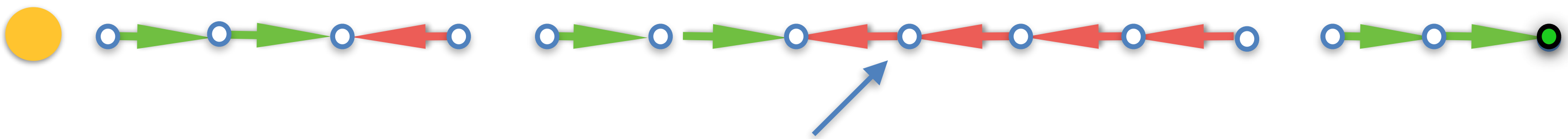


Random Listening:

- follow advice pointers with a fixed probability λ , otherwise:
- do a random walk

"Random Walks in Random Environments (RWRE)"

Example: Line graph

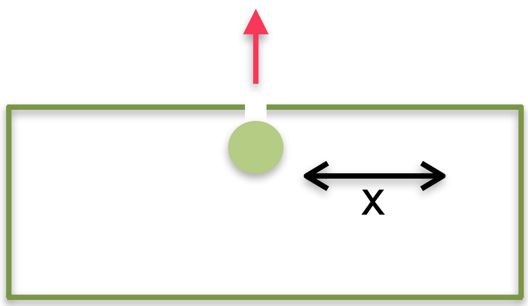


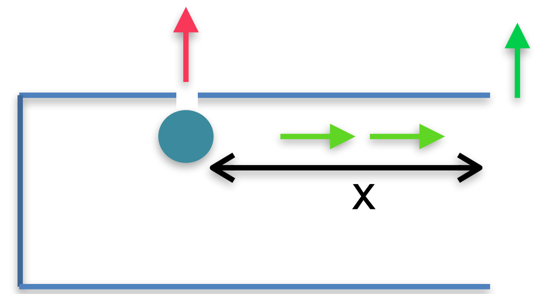
Time to pass a trap is exponential in its size

But traps are exponentially rare

Theorem: If $p>0$ then any listening probability $0< \lambda <1-p$ allows for RL to reach distance d in $O(d)$ time

Predictions

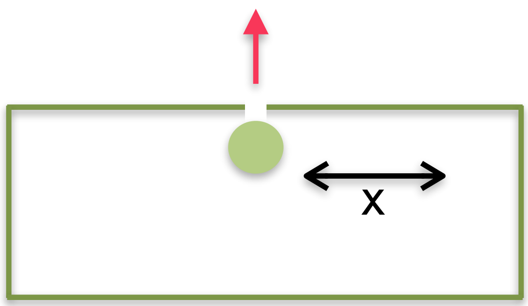


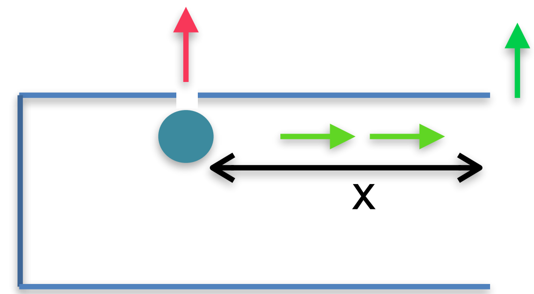


all advice is wrong ⇒
time to pass obstacle of size x is exp(x)

most advice is correct ⇒
time to pass obstacle of size x is O(x)

Predictions



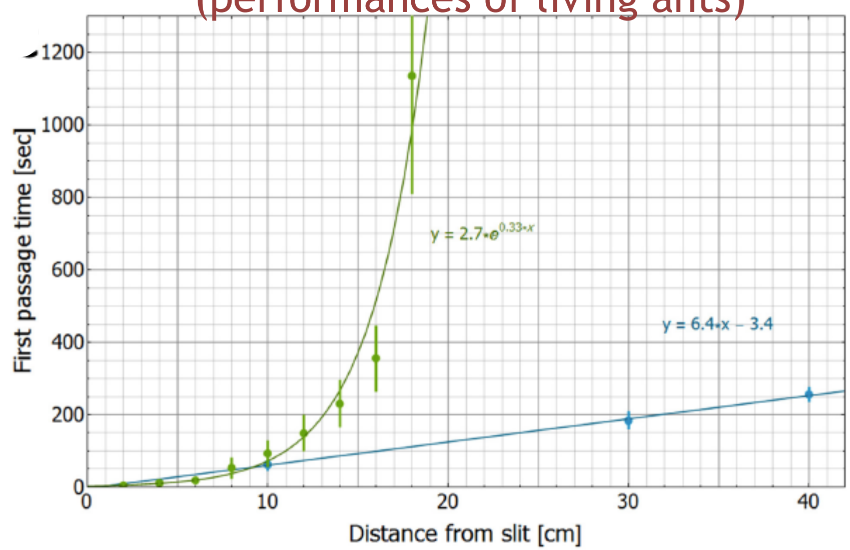


all advice is wrong ⇨
time to pass obstacle of size x is exp(x)

most advice is correct ⇨
time to pass obstacle of size x is O(x)

Experimental validations

(performances of living ants)

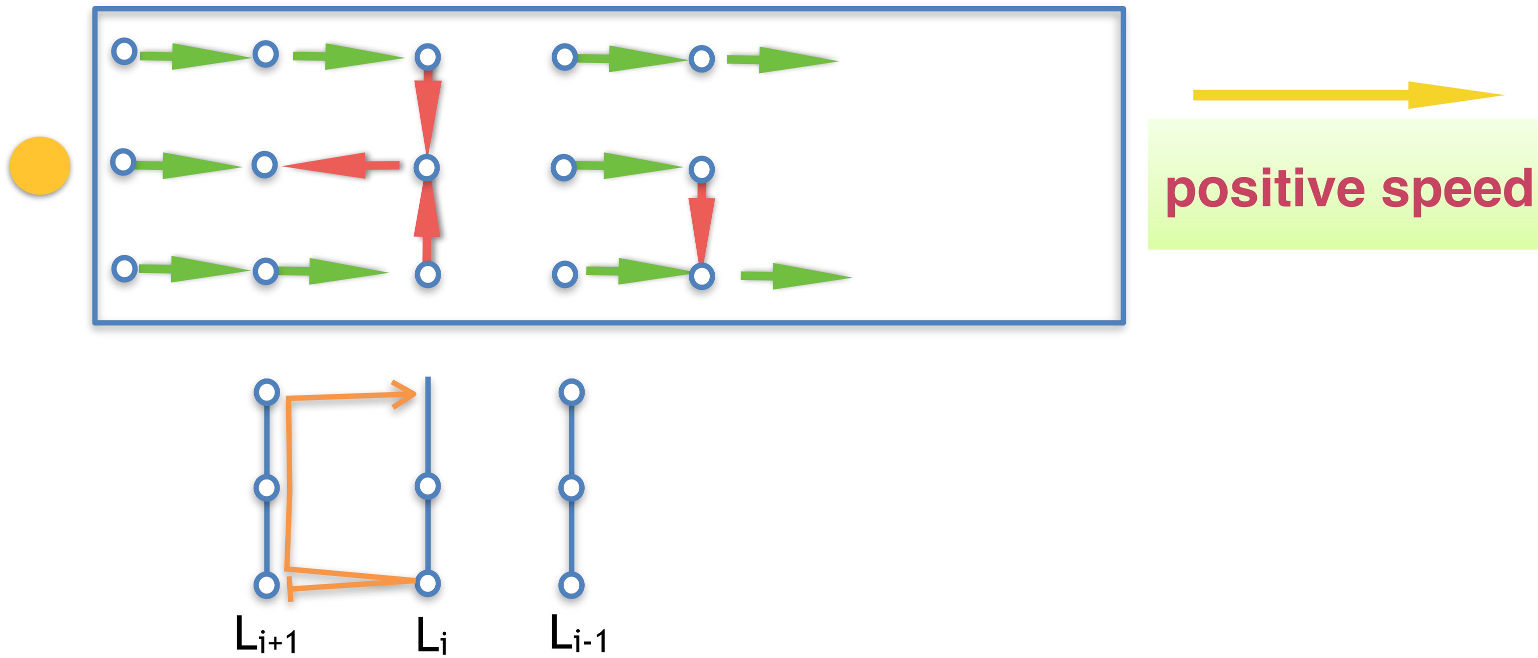


מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

E. Fonio
O. Feinerman

Consistent with step length of 10cm and probability of listening $\lambda = 0.8$

RL in grids



Situation is “in between” a line and a refreshed advice

Adapting results from RWRE [Snitzman, 2002], we prove:

Theorem:
In grids and line graph if q is small enough then there exists a range of listening probabilities to allows for positive speed

Conjecture:

There exists a constant c and a listening probability, s.t. for any graph, RL achieves linear hitting time if mistake $q < c / \Delta$ at every node.

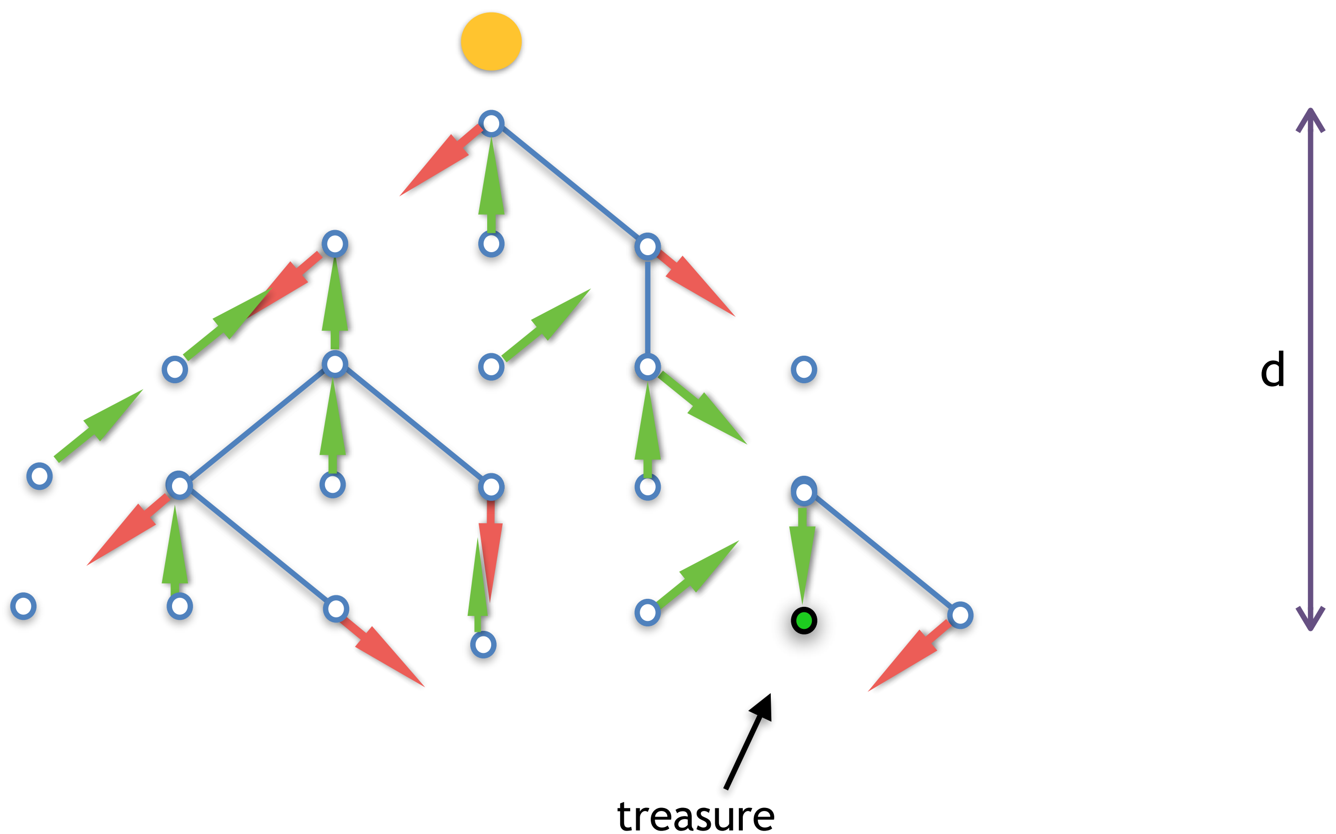
***eLife* 2016**

“Le Monde” Jan. 2017

Navigating on Noisy Trees

L. Boczkowski, A. Korman, Y. Rodeh

To be submitted



Conjecture holds for Trees

$1 / \Delta$ is a threshold for the noise q in order for random listening strategies to be efficient

Theorem

- There exists a constant c and a listening probability, s.t. for any tree, RL achieves linear hitting time O(d) if mistake $q < c/\Delta$ at every node.

- Consider the complete Δ-regular tree.
  There exists a constant c', s.t. any RL achieves exponential hitting time (in d) if mistake $q > c'/\Delta$.

Conjecture holds for Trees

$1 / \Delta$ is a threshold for the noise q in order for random listening strategies to be efficient

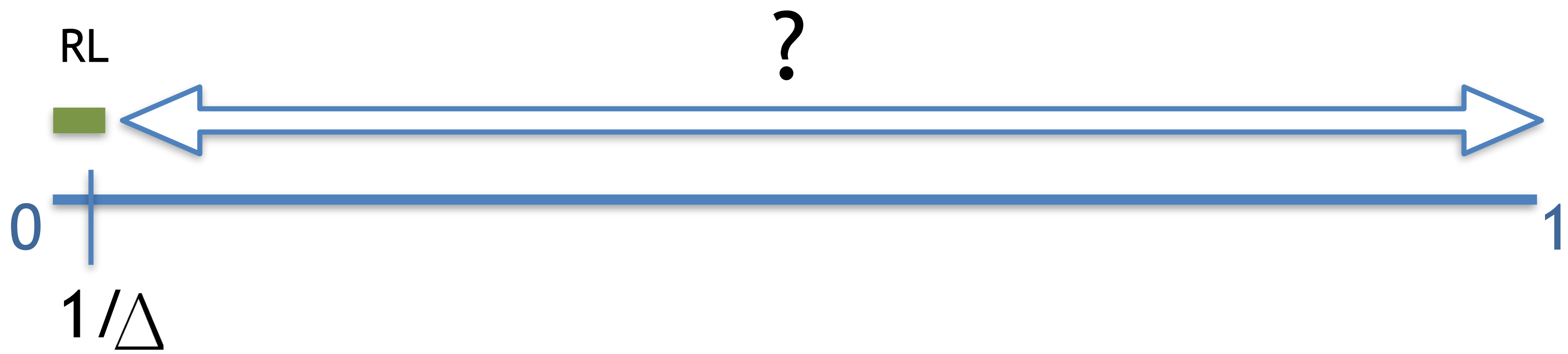
Theorem

- There exists a constant c and a listening probability, s.t. for any tree, RL achieves linear hitting time O(d) if mistake $q < c / \Delta$ at every node.

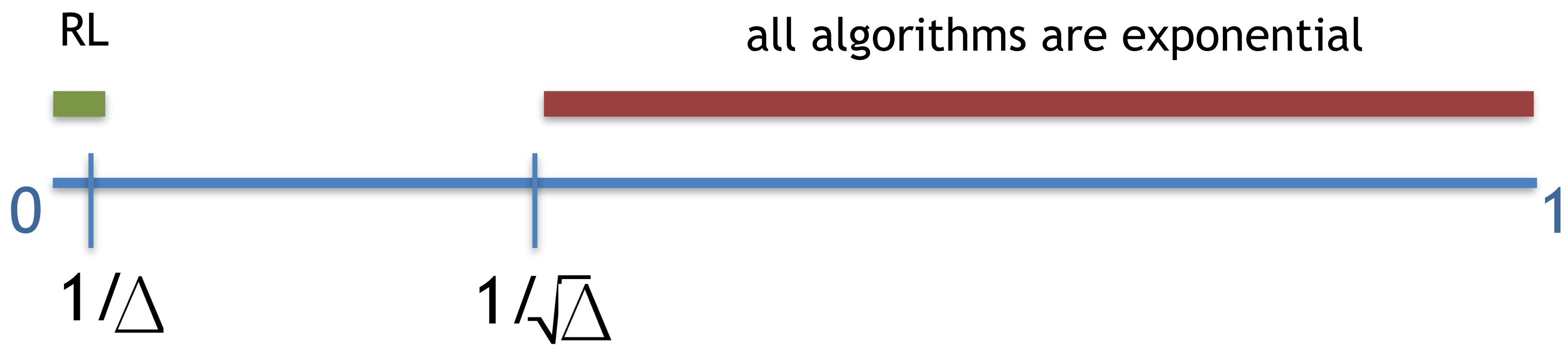
- Consider the complete Δ-regular tree.
  There exists a constant c', s.t. any RL achieves exponential hitting time (in d) if mistake $q > c' / \Delta$.

What about other algorithms?

Noise regimes

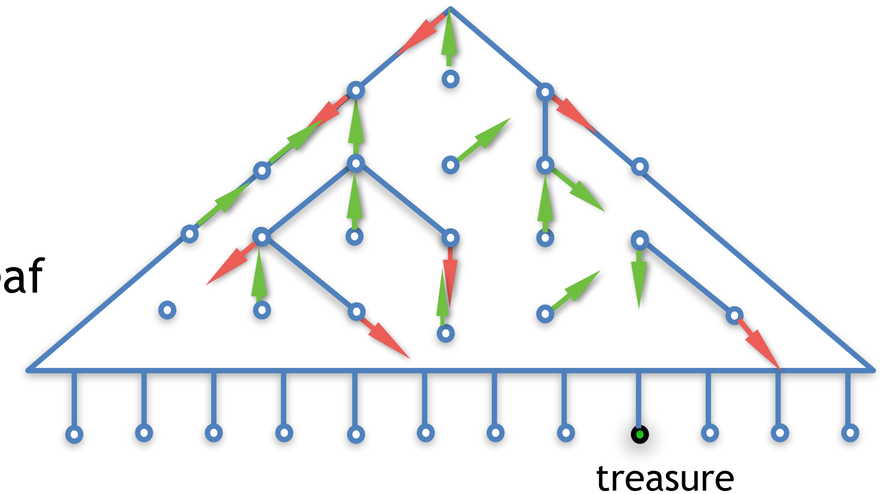


$1/\sqrt{\Delta}$ is a lower bound on noise



Proof sketch

Consider the complete Δ-ary tree



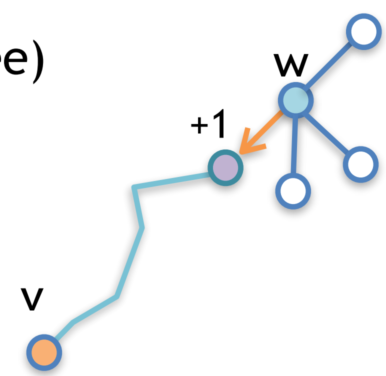
Assume:

(1) full advice is given to alg, and

(2) treasure is chosen u.a.r at a leaf

Claim:

Best algorithm: counts # of pointers (in the whole tree) pointing at each leaf and checks the leaves in order

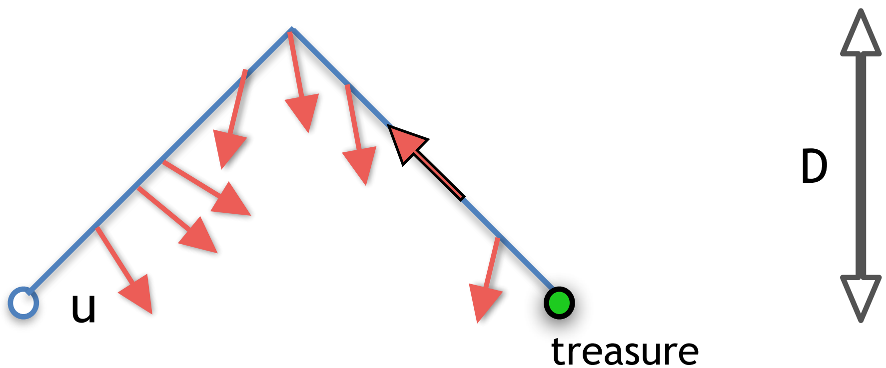


Given the claim, time is > the expected number of leaves that "beat" the treasure

Expected number of competitors

Prob that a given leaf is "better looking" than t is small, but there are many leaves!

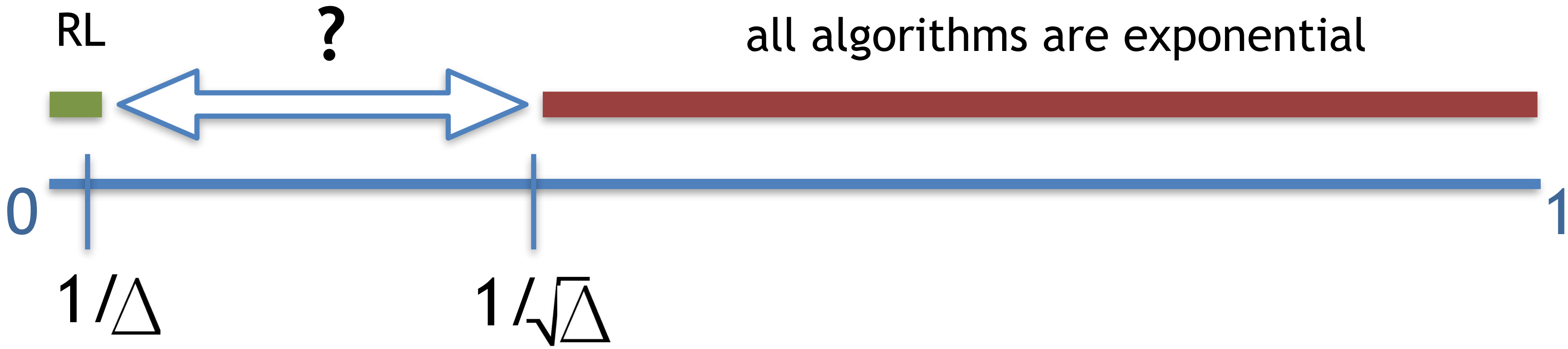
Prob that u is "better" than t is at least $\frac{q}{\Delta} \cdot q^{d(u,\tau)} \approx q^{2D}$



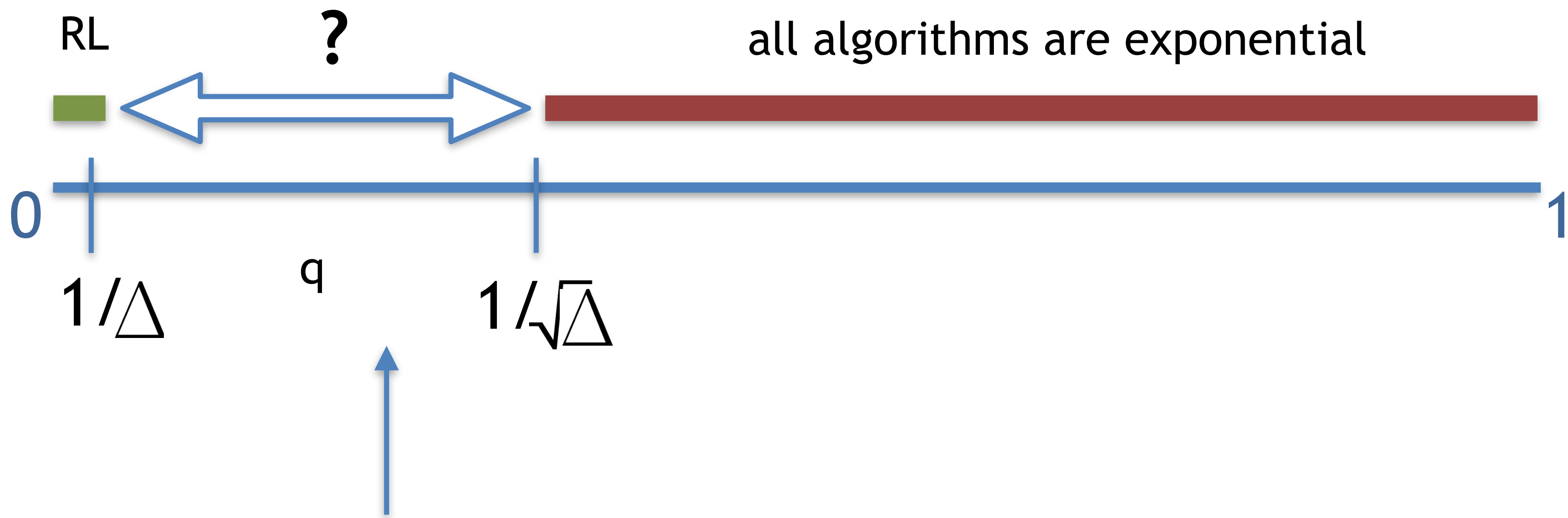
There are roughly Δ^D leaves whose distance from t is 2D

Therefore, the expected #leaves that beat t is at least: $(q^2\Delta)^D$

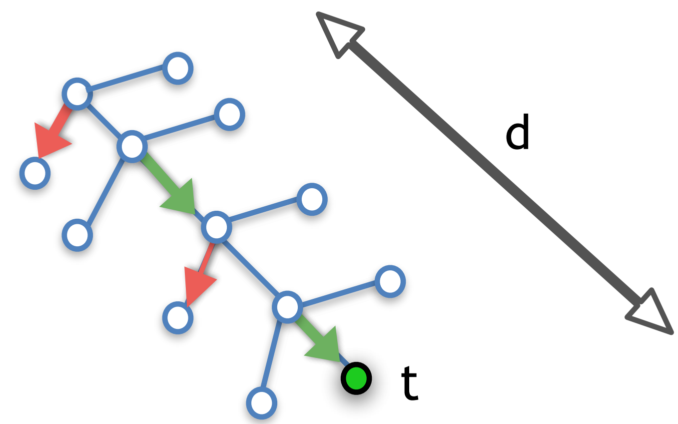
Noise regimes



Noise regimes



Time lower bound of $\Omega(dq\Delta) = \Omega(d\sqrt{\Delta})$



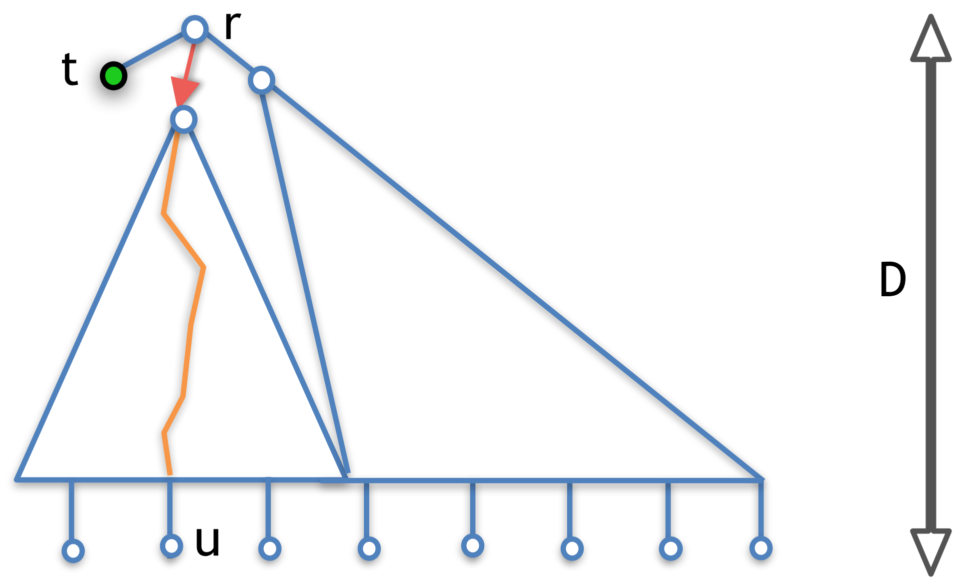
A simple greedy strategy that fails

Walk to the currently most promising node - the one with most pointers pointing to it

A simple greedy strategy that fails

Walk to the currently most promising node - the one with most pointers pointing to it

Consider the complete Δ-ary tree with an extra child to r



There are $(\Delta - 1)^D$ leaves u. For any of them, prob that r points to u, and nobody on the path to u points at r is > $\frac{q}{\Delta} \cdot q^{D-1} \approx q^{D-1}$

So the expected number of nodes visited before the treasure is at least roughly:

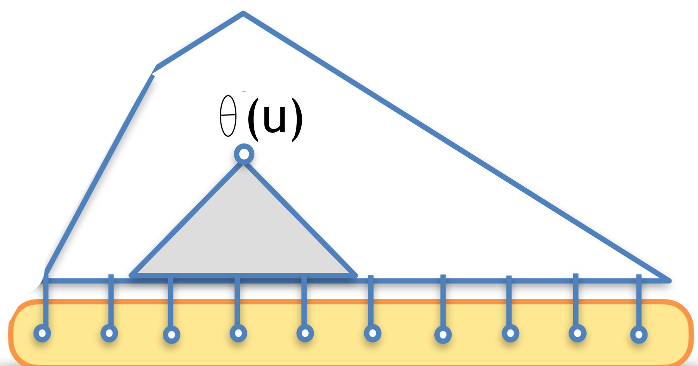
$$(q\Delta)^D$$

Theorem: if $q<1/\sqrt{\Delta}$ then there exists a walking algorithm that runs in optimal $O(d\sqrt{\Delta})$ time

Intuition for the construction: Based on a Bayesian approach

Let us make our life easier, and assume:

1. tree structure is known to the algorithm
2. treasure is restricted to leaves
3. t is chosen at random according to a known dist θ



A prior distribution θ

Algorithm: go to the node on the border of what you saw that maximizes the prob that the treasure is a descendant of that node

We want to choose θ s.t. the corresponding algorithm will be good against an adversary

So what would be the good choice of θ ?

The most natural choice is the uniform over all leaves or over all nodes.

This works for compete Δ-ary trees, but fails for general trees :-(

Choosing θ

We define θ according to a random walking down process:

Starting at the root, walk down to a child u.a.r. until reaching a leaf

For a leaf v, define θ(v) as the probability that this process eventually reaches v. Our extension of θ can be interpreted as θ(v) being the probability that this process passes through v. Formally,

$$\theta(\sigma) = 1, \text{ and } \theta(u) = 1/\prod_{w\in[\sigma,u\rangle} \Delta_w$$

Choosing θ

We define θ according to a random walking down process:

Starting at the root, walk down to a child u.a.r. until reaching a leaf

For a leaf v, define θ(v) as the probability that this process eventually reaches v. Our extension of θ can be interpreted as θ(v) being the probability that this process passes through v. Formally,

$$\theta(\sigma) = 1, \text{ and } \theta(u) = 1/\prod_{w\in[\sigma,u\rangle} \Delta_w$$

This works! it gives an algorithm that runs in $O(d\sqrt{\Delta})$ time

The optimal walking algorithm

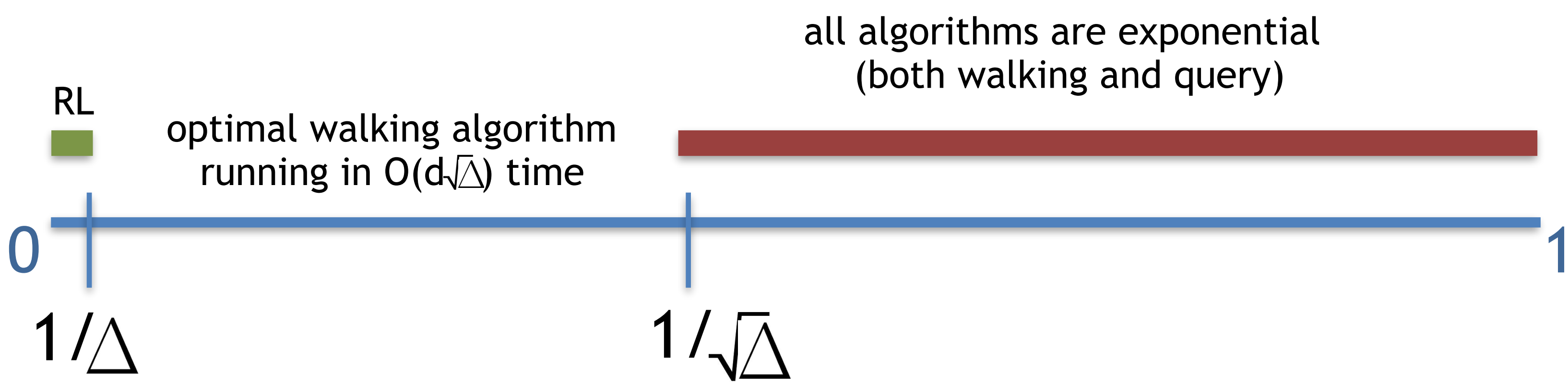
$\theta(\sigma) = 1$, and $\theta(u) = 1/\prod_{w \in [\sigma, u\rangle} \Delta_w$

$$\mathtt{score}(u) = \frac{2}{3}\log(\theta(u)) - \sum_{w \in \overleftarrow{\mathbf{adv}}(u)} \log(\Delta_w)$$

Consider all advice discovered so far, and go to a node on the border with highest score

Note, the algorithm does not need any a priori knowledge of the structure of the tree!

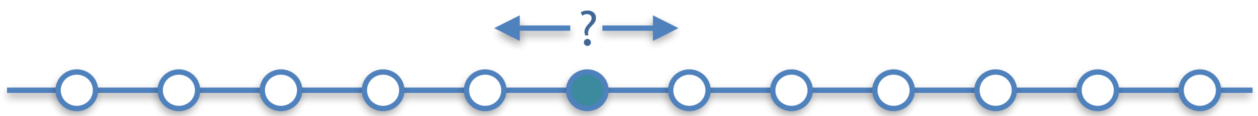
Query Algorithms



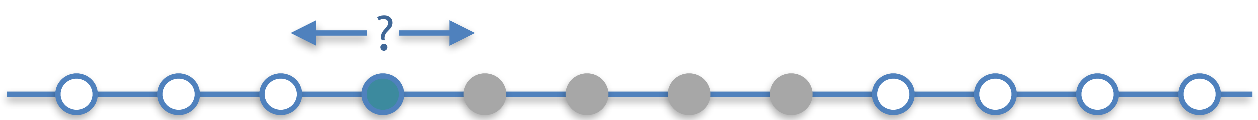
What about query algorithms?

O(log n) query algorithm for the line

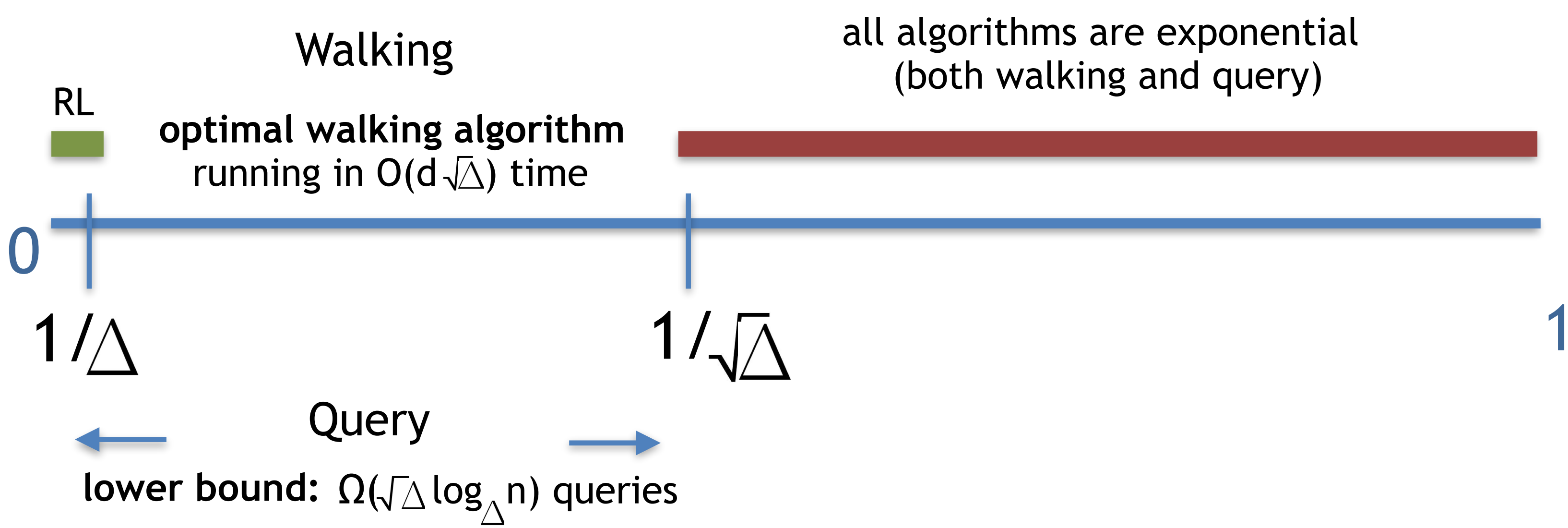
In the case of refreshable noise, there exists an O(log n) query algorithm [Feige et al. SICOMP 1994]



It is easy to immolate any protocol on the line by simply querying one of the neighbors of endnotes of the corresponding visited subpath



Theorem: Lower Bound of $\Omega(\sqrt{\Delta} \cdot \log_{\Delta} n)$ for complete Δ-ary trees



Query algorithms

Theorem: There is a query algorithm with # of queries $O(\sqrt{\Delta} \cdot \log^2 n)$ on expectation, when $q < 1/\sqrt{\Delta}$

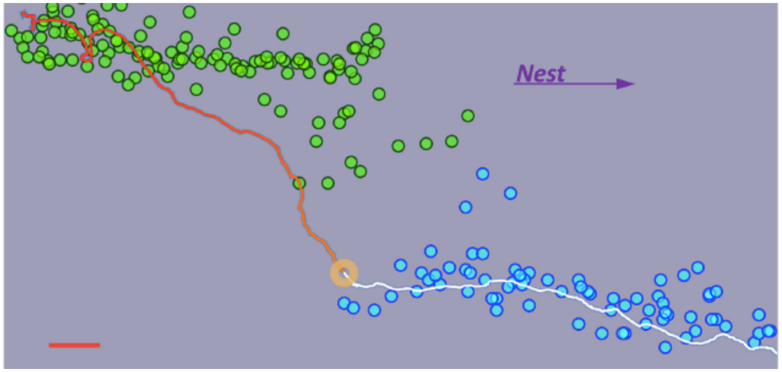
Basic strategy

- Do a separator decomposition.
- For each junction v of the sep tree, apply the walking alg on the subtree T_v of depth $O(\log n)$, until finding a leaf w of T_v, for which 80% of the arrows point to it. W.h.p., this will happen by time $O(\sqrt{\Delta} \log n)$.
- Once w is point, the neighbor of v in the sep tree that contains w in its tree is w.h.p the correct separator to continue.
- W.h.p. this finds the treasure within $O(\sqrt{\Delta} \log^2 n)$. If the treasure is not found by this time, do an exhaustive search.

Theorem: There is a query algorithm with # of queries $O(\sqrt{\Delta} \cdot \log n \cdot \log\log n)$ for Δ-regular trees

Summary

A new kind of ant trail



A new kind of model for search in unreliable conditions

RL - A memoryless strategy that is good for
grids and trees as long as $q<c/\Delta$

Conjecture: RL is good for any graph as long as $q<c/\Delta$

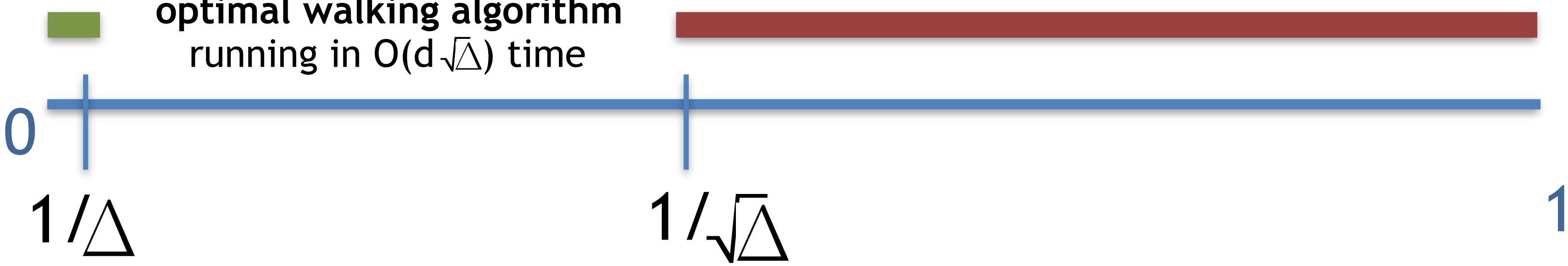
Summary - on noisy trees

Walking

RL

optimal walking algorithm
running in $O(d\sqrt{\Delta})$ time

all algorithms are exponential
(both walking and query)



Query

lower bound: $\Omega(\sqrt{\Delta}\log_{\Delta} n)$ queries

upper bounds

1. query algorithm
running in $O(\sqrt{\Delta}\log^2 n)$ time

2. query algorithm
running in $O(\sqrt{\Delta}\log n \log\log n)$ time
for Δ-regular trees

Open problems

- Solve the random listening conjecture for general graphs
- Find optimal algorithms for other graph families (e.g. expanders?)

Merci!