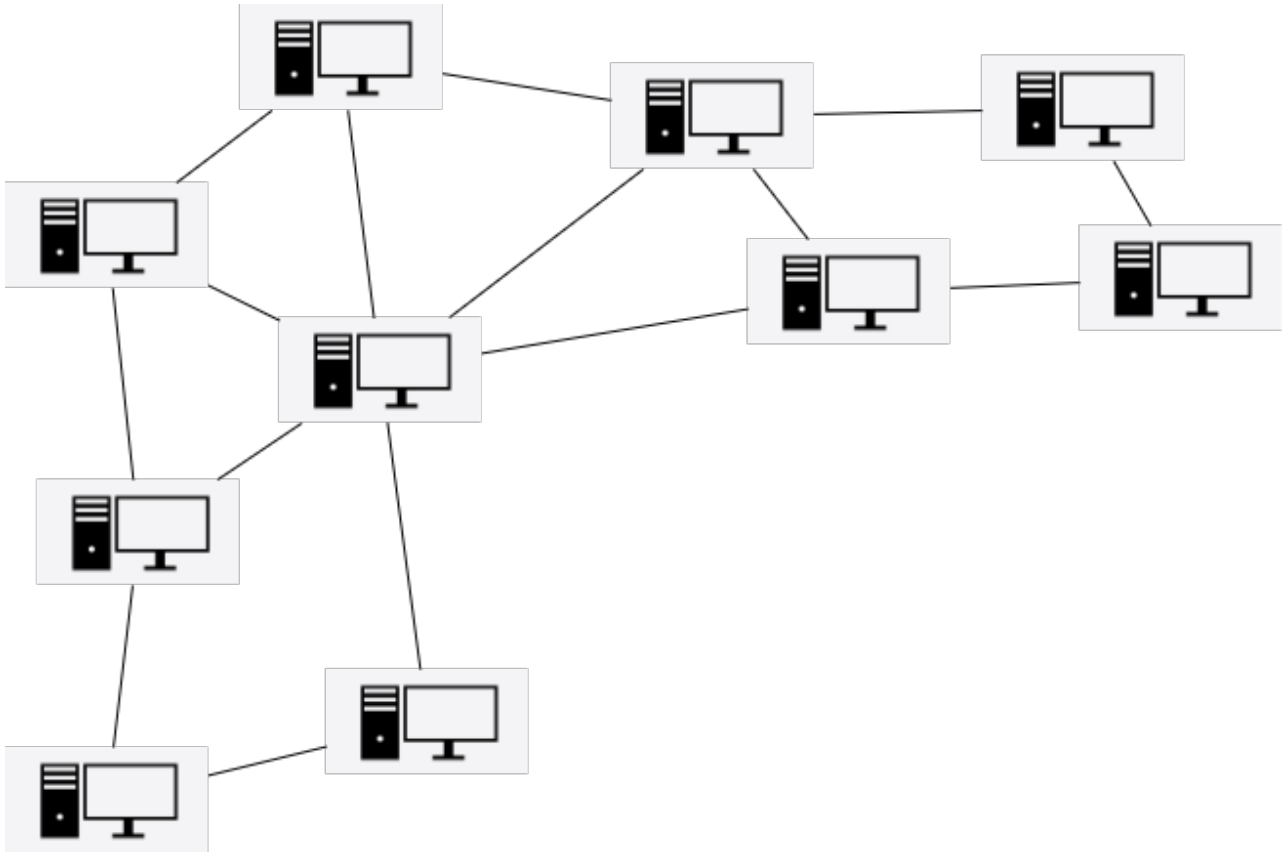
LCLs with and without knowledge of n

Gustav Schmid

The LOCAL Model

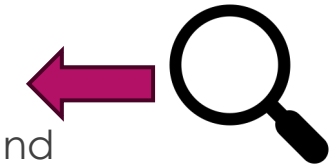


- Nodes have unbounded computation
- Arbitrarily large messages to all neighbors
- Measure: #communication rounds

Called LOCAL because in T rounds nodes only learn the local T-hop information

The LOCAL Model (with extras)

- Nodes also know the value n
  - Sometimes a linear upper bound
  - Sometimes a polynomial upper bound
- Nodes have Ids (typically at most n^c)
- Nodes often know the max deg. Δ
- The graph class (e.g. trees/grids/planar/…)
- …



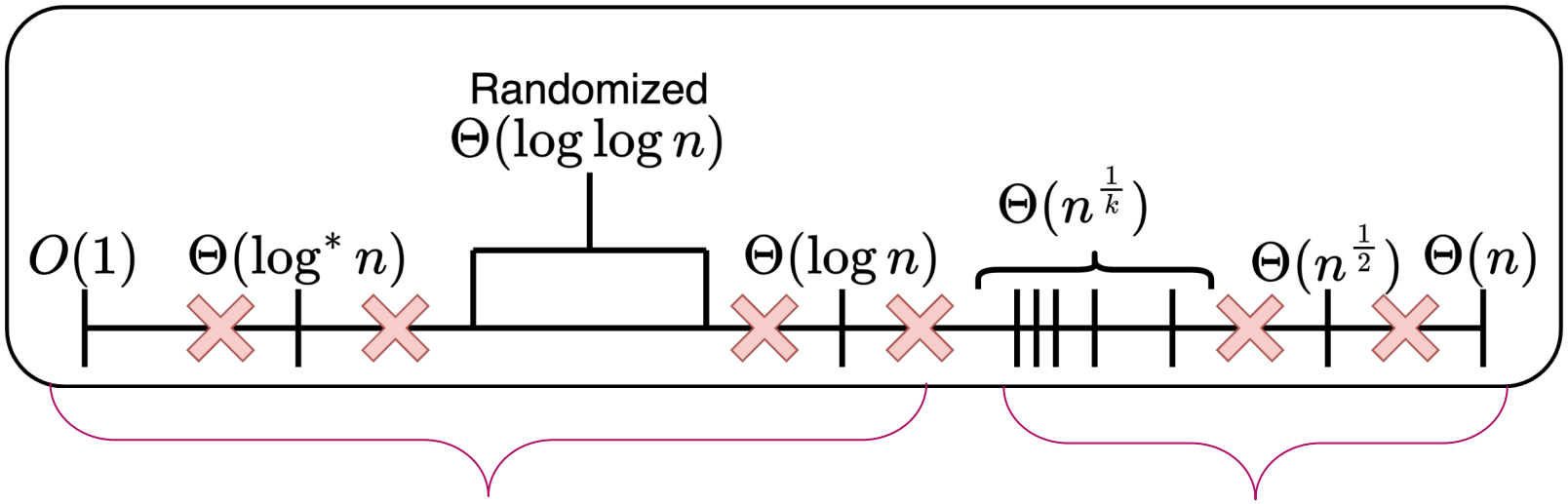
Does not feel very LOCAL!

Just convenience?

Based on work with Alkida, Dennis, Sebastian, Fabian and Timothe Picavet [PODC 26]

Case Study – LCLs in Trees

Landscape of LCLs on bounded degree Trees



Lower Regime

Poly Regime

- Everything works without knowing n
- Largely because we focus on topology!

[CKP, FOCS16], [Chang&Pettie, FOCS17], [BCGGRV, ITCS22]

LCLs

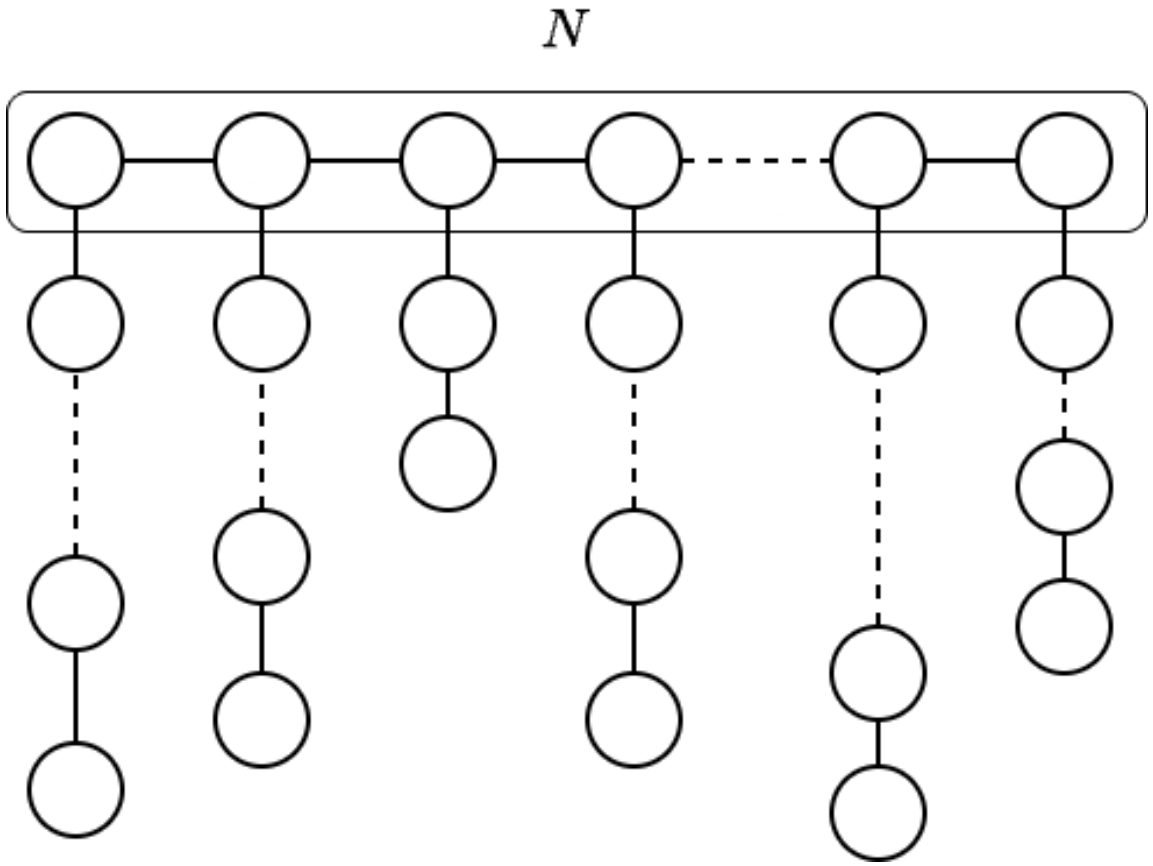
- Labelings that can be verified within constant radius
- Coloring/MIS/Matching/…
- In bounded degree trees

What happens to this polynomial regime when nodes don't know n?

Two Problems – 2.5 coloring (simplified)

2-color this

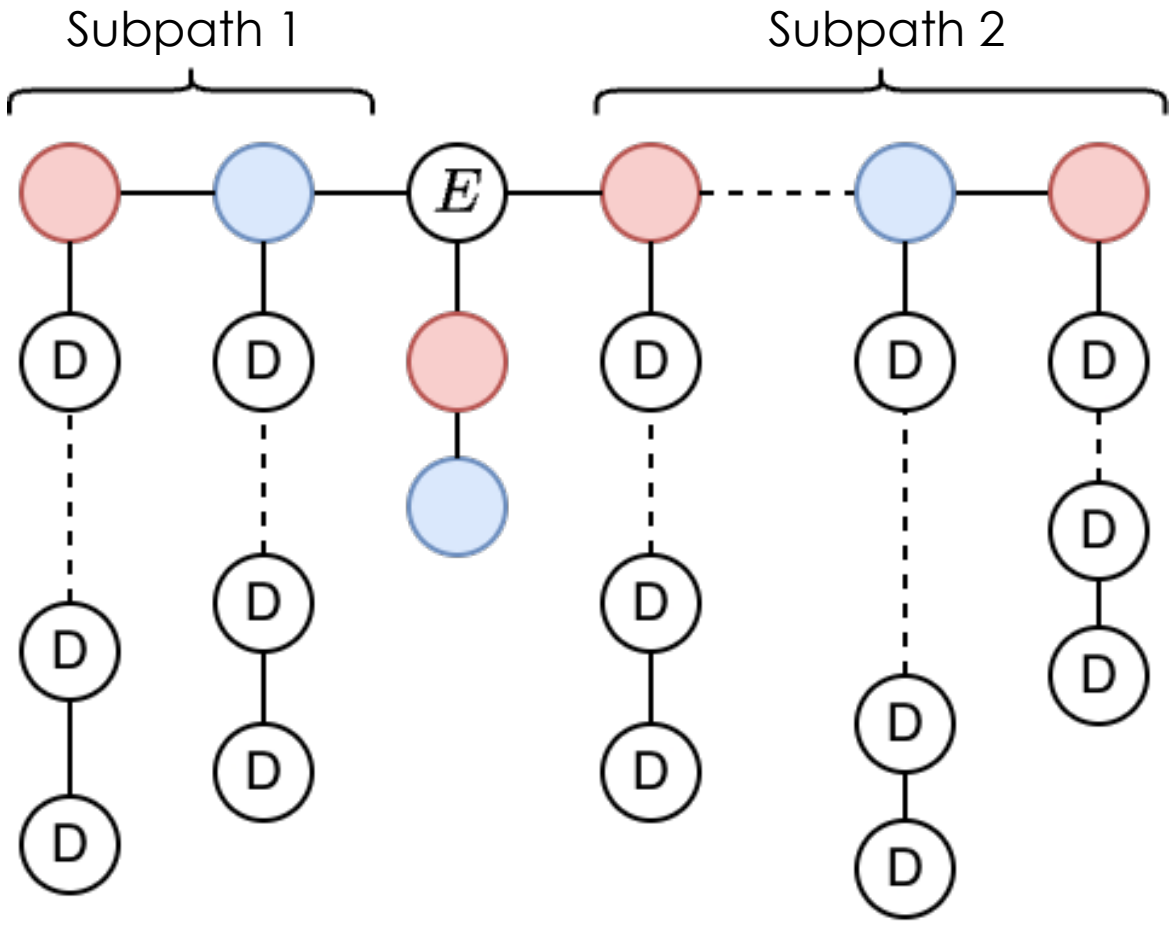
Needs $\Omega(N)$ rounds



The Problem – Bottom Paths

2-coloring bottom paths splits top path

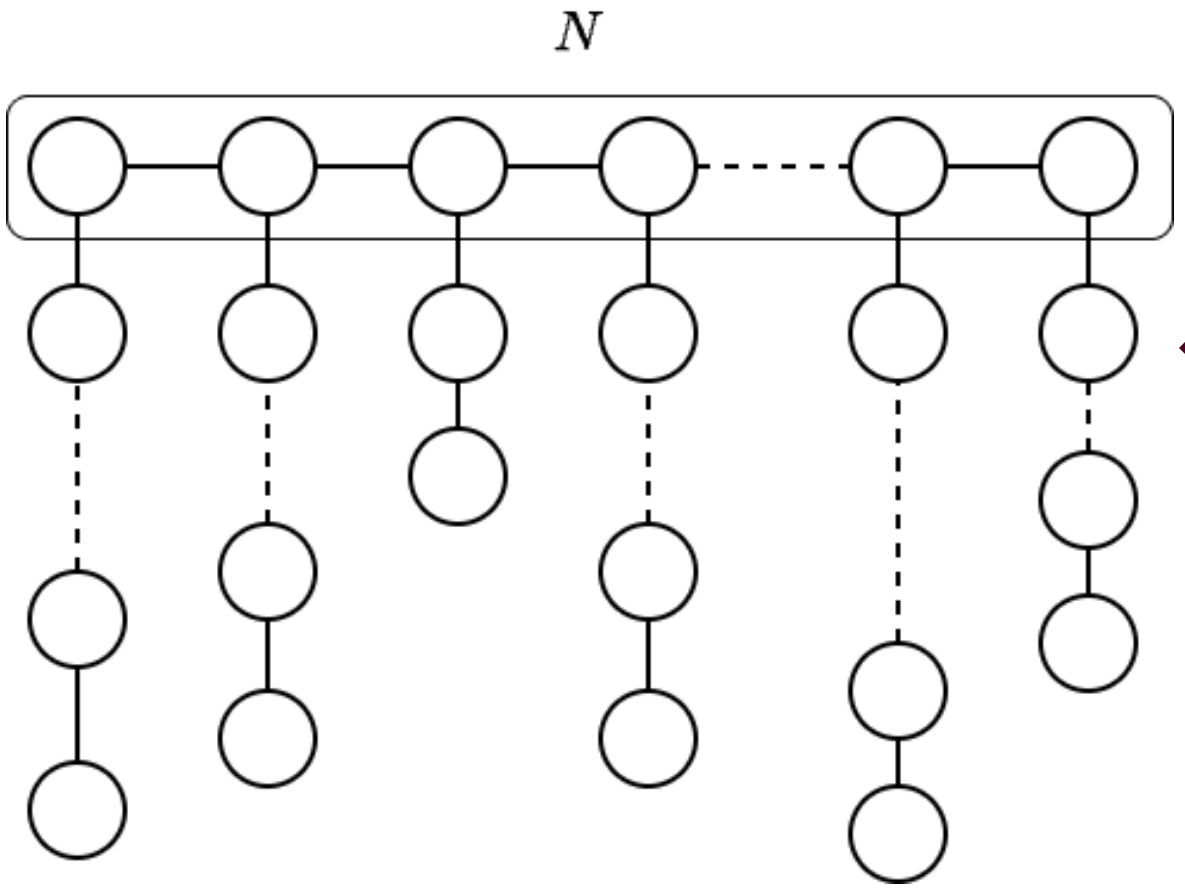
If we don't 2-color we need to *Decline*



The Problem

2-color this

Needs $\Omega(N)$ rounds



Split into subpaths by 2-coloring bottom

Needs L rounds
If bottom path has length L

Upperbound – LOCAL algorithm

Bottom Nodes

- Check lengths of bottom path
- Length < L → we 2-color
- Length ≥ L → we Decline

Needs L rounds

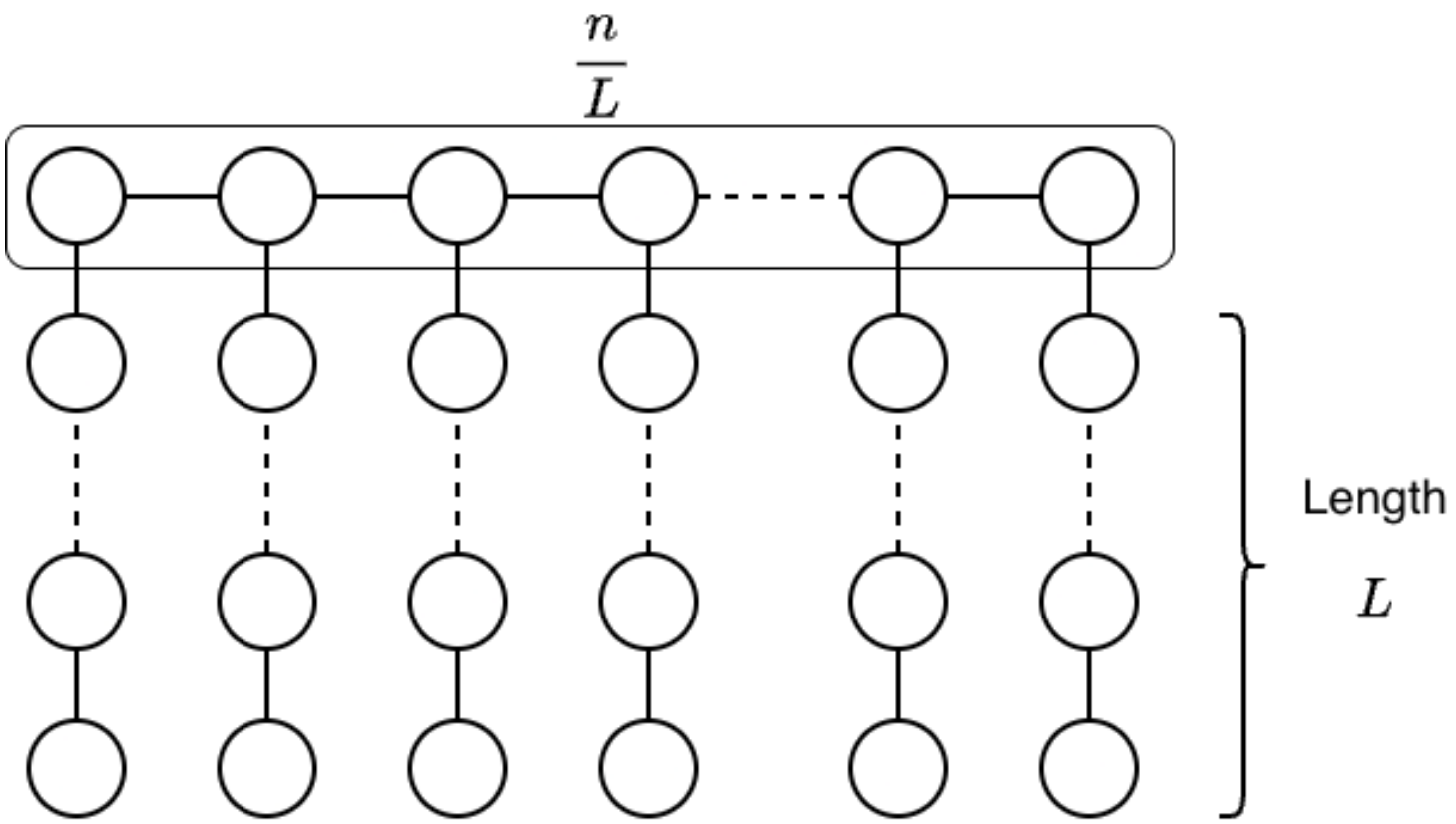
Top Nodes

- Wait L rounds
- If bottom is colored → Output Exempt
- 2-color all remaining nodes
- Only $\frac{n}{L}$ top level nodes are not Exempt

Needs $\frac{n}{L}$ rounds

Overall $O(\sqrt{n})$

Lower Bound



We need to 2-color at least one path

$\Omega\left(\min\left\{L, \frac{n}{L}\right\}\right)$

$\Omega(\sqrt{n})$

Knowledge about n?

Upperbound: "set L = $\sqrt{n}$"

Implicitly needs nodes to know n

Is this just an artefact of our algorithm design?

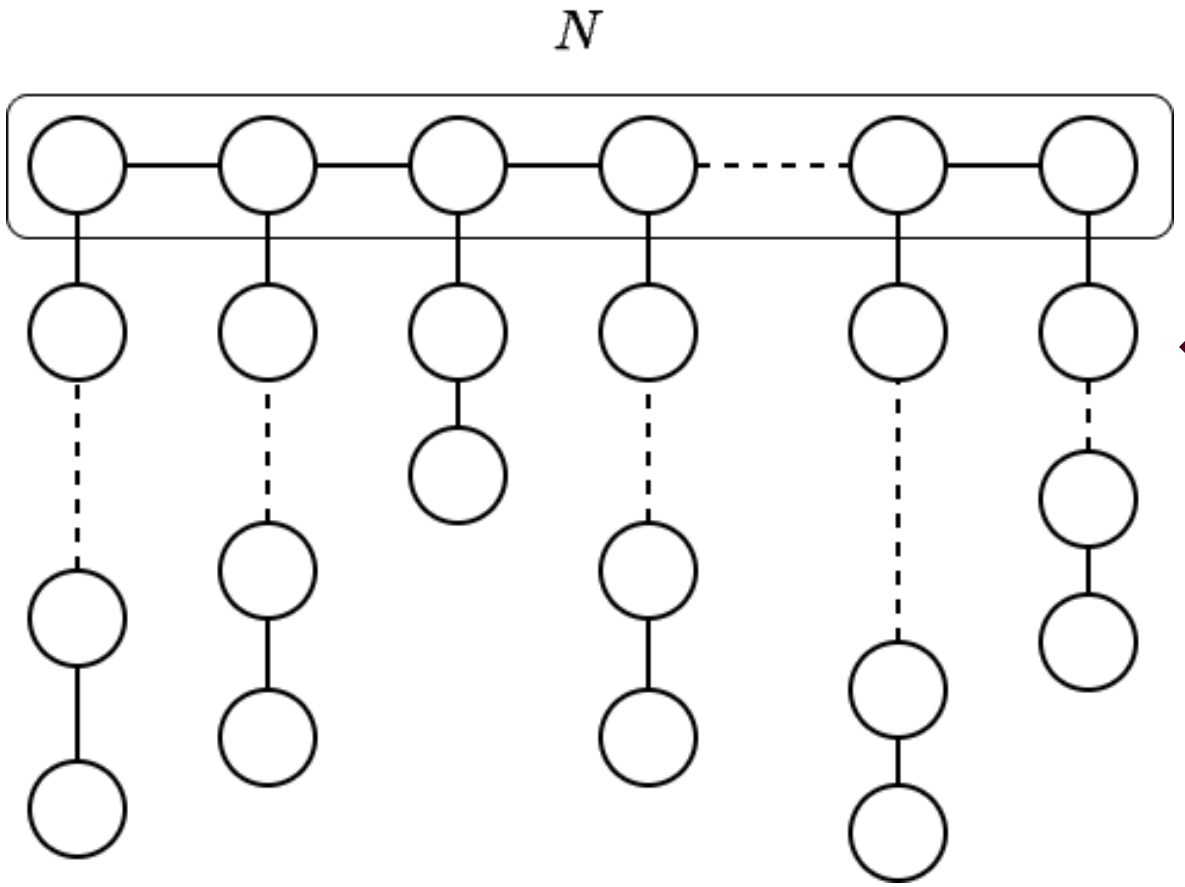
What if nodes did not know n?

No Knowledge LOCAL

- Same Problem
- No initial knowledge about n
- Deterministic
- Unbounded IDs

No Knowledge

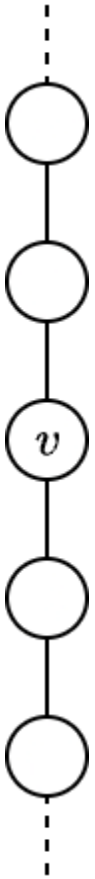
2-color this



Split into subpaths by 2-coloring bottom

Now nodes don't know n

Bottom Perspective



- Path of bottom (level 1) nodes
- v sees only a path
- IDs are ridiculously large
- No clue what n is supposed to be
- We will never show v the end of the path

How many rounds till v declines?

We get $\Omega(n)$ for all $O\left(n^{\frac{1}{k}}\right)$ problems

Possibility 1: v never declines

Then the instance is only this bottom path $\rightarrow \Omega(n)$

Possibility 2: v declines after a constant C number of rounds

Top path of length $\frac{n}{c}$, bottom paths decline $\rightarrow \Omega(n)$

So?

- The problem depends on this initial knowledge!
- If we make an initial assumption, what should it be?

Try Polynomial Upperbound

- Nodes are given two values N, c
- Promise that $\mathrm{n} \leq N \leq n^c$

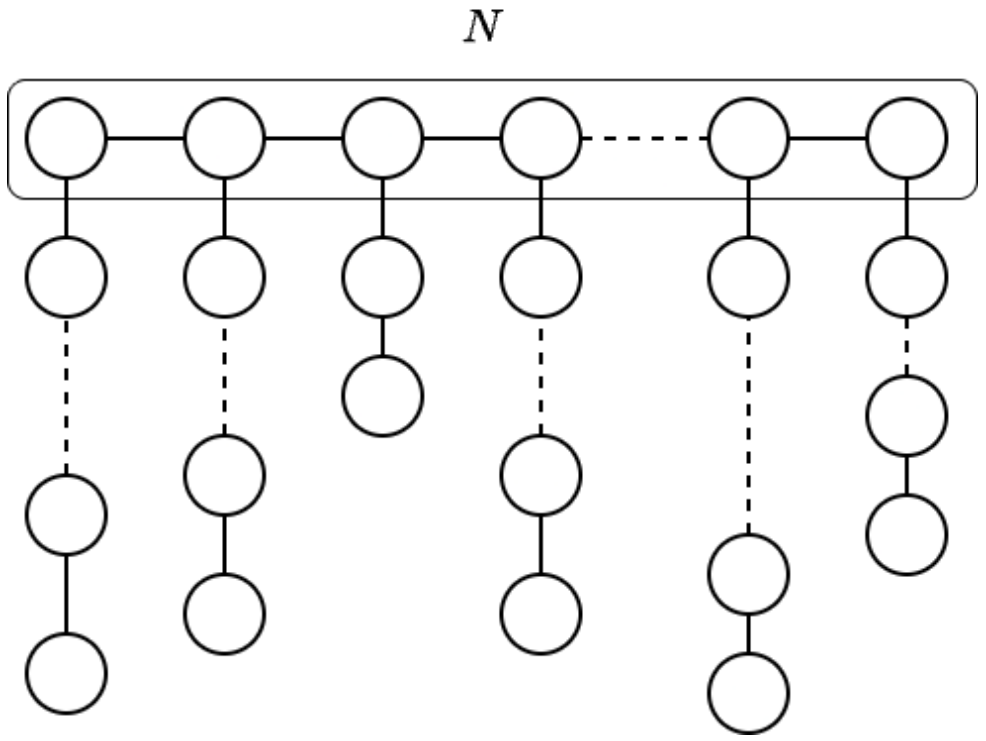
Polynomial Upperbound given

We can adapt the original algorithm

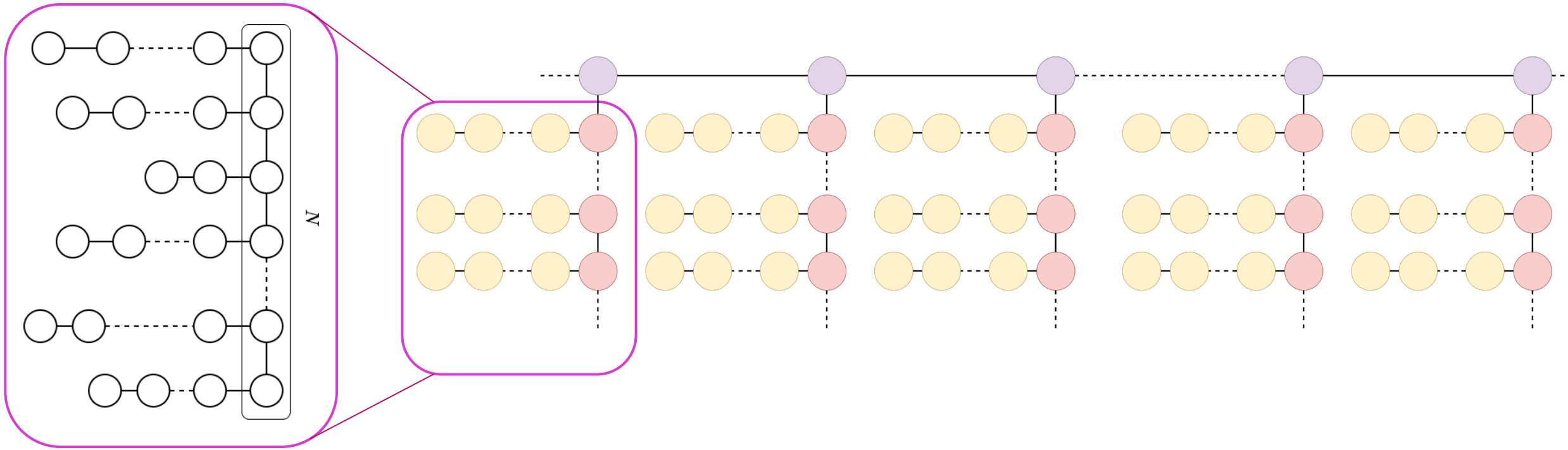
- For k = 2 this works → $\Theta\left(n^{\frac{c}{c+1}}\right)$

Can't prove lower bound for k > 2

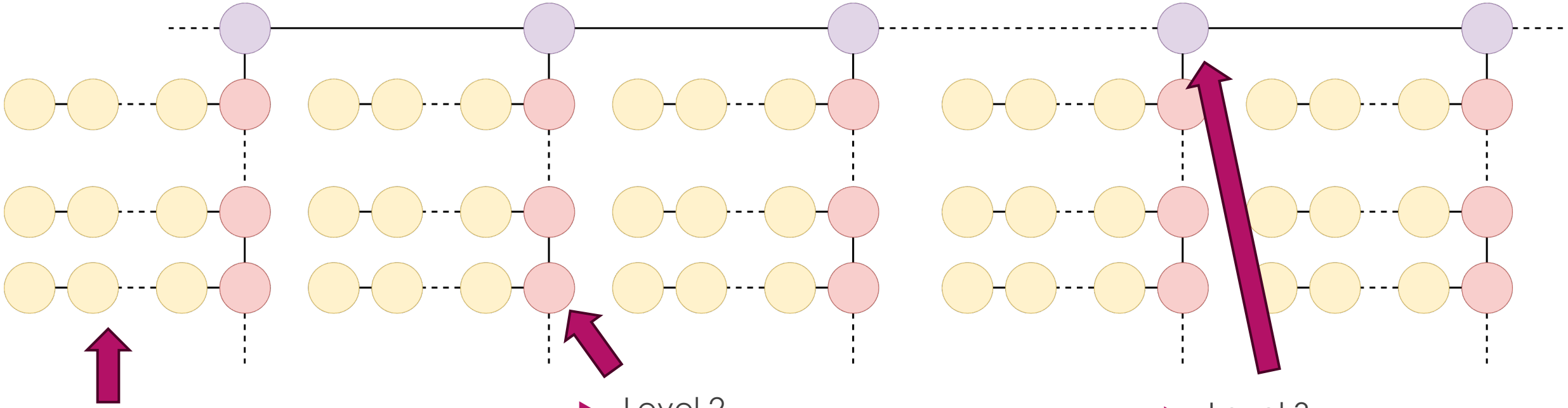
The Problem – with more levels



The Problem – with more levels



The Problem – with more levels



- Level 1
- 2-color or decline

- Level 2
- Exempt if possible
- 2-color or decline subpaths

- Level 3
- Exempt if possible
- 2-color subpaths

Polynomial Upperbound given

We can adapt the original algorithm

- For k = 2 this works → $\Theta\left(n^{\frac{c}{c+1}}\right)$

Can't prove lower bound for k > 2

- For k > 2 the problem is recursive
  - We have k-levels
  - Each level is a collection of paths we 2-color/Decline
  - 2-color level 1 paths of length L → Remaining instance (level 2+) of size $n' = \frac{n}{L}$
  - 2-color level 2 paths of length L → Remaining instance (level 3+) of size $n'' = \frac{n'}{L}$
  - …
  - 2-color remaining paths in level k. $O\left(\frac{n}{L^{k-1}}\right)$

If we know n

$L = n^{\frac{1}{k}} \rightarrow \Theta\left(n^{\frac{1}{k}}\right)$

The Intuition – Polynomial upper bound

- We have to choose L as a function of N and c!
- $L = N^{\alpha}$, where α depends on c
- Runtime of level 1, 2, 3, 4, … : $N^{\alpha} \sim n^{c\alpha}$ ⟸ Then this can't happen for level 2, $n' < N^{\alpha}$
- Size of the next level: $n' = \frac{n}{N^{\alpha}} \sim \frac{n}{n^{\alpha}}$

Actually, if $N \sim n^{c}$ size decreases a lot!

Give each level i its own $L = N^{\alpha_i}$
make the α values progressively larger

- Optimization problem gives $\alpha_1, \dots, \alpha_{k-1}$
- Complexity $O(n^{c\alpha_1})$

It's tight! We prove $\Omega(n^{c\alpha_1})$

Polynomial Upper bound - Summary

- Setting: nodes are given N, c such that $n \leq N \leq n^c$
- 2.5-coloring has complexity $\Theta(n^{c\alpha})$
- α is from an optimization problem

What about other problems in the $O(n^{1/k})$ classes?

- The problem k-R&C is $O(n^{1/k})$ -complete
- If problem P is in $O(n^{1/k})$, then P can be solved by solving k-R&C

We show k-R&C also has complexity $\Theta(n^{c\alpha})$

The optimization problem exactly captures the difficulties of this setting!

Summary (So far)

2 Problems, the canonical 2.5-coloring and k-R&C, which is **complete** for the classes $O\left(n^{\frac{1}{k}}\right)$

No Knowledge

- Nodes do not know anything about n
- Unbounded Ids
- K-R&C is $\Omega(n)$ for all k

Polynomial Promise

- Nodes are given N,c with promise $n \leq N \leq n^c$
- Exploit some tradeoff
- Optimization problem gives tight results
- Get $\Theta(n^{\alpha c})$ for k-R&C

Open Question:

Other polynomial problems?

Do all Polynomial Problems follow this tradeoff?

Polynomial Upper bound - IDs

- What about IDs?
- Algorithm uses them only for symmetry breaking

What are reasonable assumptions about the ID space?

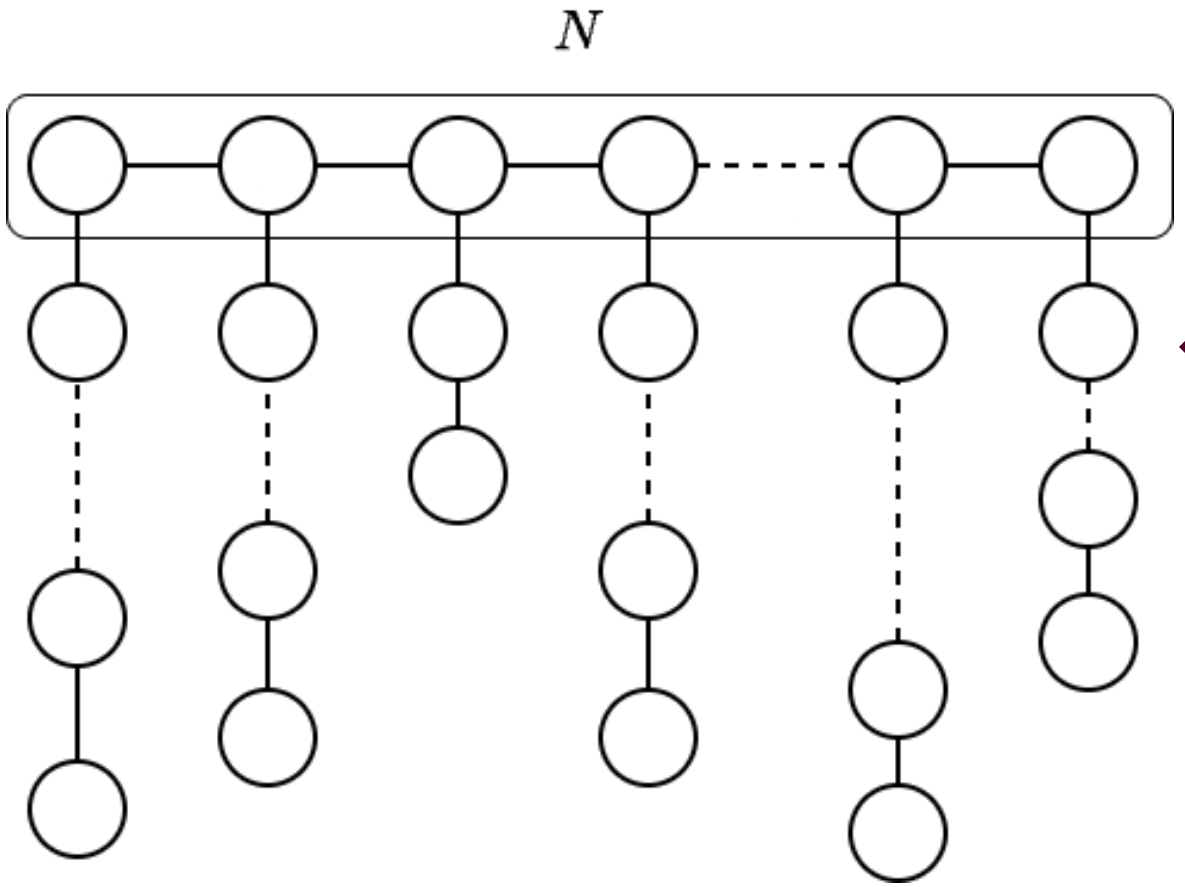
- Setting: nodes are given only c, such that $ID \leq n^c$
- Nodes know nothing (else) about n
- Our lowerbounds already hold in this setting

For 2.5-coloring we give the same upperbound $O(n^{c\alpha_1})$!

Only Ids

2-color this

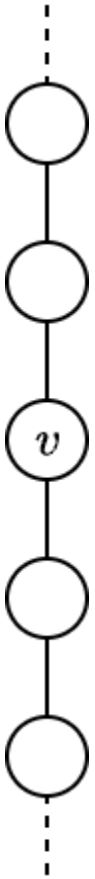
Remember for k=2 this is the problem.



Split into subpaths by 2-coloring bottom

Nodes do not know n
Ids are in O(n).

Only Ids - Intuition



- Bottom node v sees only a path
- No clue what n is supposed to be
- If we see the entire path, we can 2-color

How many rounds till v declines?

Let ID_t be the largest ID observed by v in t rounds

v declines if $\sqrt{ID_t} < t$

Consequences:

All bottom paths decline after at most $\sqrt{n}$ rounds

Some bottom paths decline almost immediately!

But there are only roughly L paths that decline within L rounds.

We get $O(\sqrt{n})$ again!

Only Ids

- This generalizes to any k and c!

$\forall\, k, c \rightarrow T_{2.5}(n)$ remains the same only using promise on the Ids!

- We prove this only for 2.5-coloring, but we think it applies to other problems aswell:

Conjecture: Any LCL on trees has the same asymptotic complexity, if only a promise on the size of the largest Id is given.

Other polynomial problems?

Summary

Thanks, Questions?

2 Problems, the canonical 2.5-coloring and k-R&C, which is **complete** for the classes $O\left(n^{\frac{1}{k}}\right)$

No Knowledge

- Nodes do not know anything about n
- Unbounded Ids
- K-R&C is $\Omega(n)$ for all k

Randomization?

- Helps in a non-trivial way (unexpectedly)
- 'estimate log n'

Polynomial Promise

- Nodes are given N,c with promise $n \leq N \leq n^c$
- Exploit some tradeoff
- Optimization problem gives tight results
- Get $\Theta(n^{\alpha c})$ for k-R&C

Polynomial Promise on Ids

- Nodes are just c with the promise that Ids $\leq n^c$
- This is just as good for 2.5-coloring, we get $\Theta(n^{\alpha c})$
- This very likely extends to all problems

So?

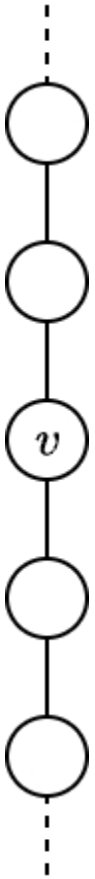
- The problem depends on this initial knowledge!
- If we make an initial assumption what should it be?

Randomization?

- No knowledge about n
- Unlimited Random Bits (no lds)
- This usually does not help for polynomial problems ($O(n^{1/k})$ remains $O(n^{1/k})$)

Surprisingly, it does help! We get $O\left(\frac{n}{\log n}\right)$

k = 2 Bottom Perspective (Randomized)



- Path of bottom (level 1) nodes
- v sees only a path
- No clue what n is supposed to be

How many rounds till v declines?

Mark a node with prob. $\frac{1}{2}$
v declines if it sees a marked node

We get $O\left(\frac{n}{\log n}\right)$

Consequences:

v declines in $O(\log n)$ rounds w.h.p.

A $o(\log n)$ path 2-colors with prob. $n^{-\varepsilon}$ for very small ε

Top paths become poly(n) length, only if most bottom paths have length $\Omega(\log n)$

No Knowledge - Randomized

- For k = 2, we get $O\left(\frac{n}{\log n}\right)$

Randomization helps in a non-trivial way to estimate $\log n$

Again, this is tight, we prove $\Omega\left(\frac{n}{\log n}\right)$ lowerbound

For larger k, we use level 1 paths to get an estimate of log n, then try and abuse this information

This gives us weird complexities. For k=3, we get $\tilde{\Theta}\left(\frac{n}{f(n)}\right)$, where $f(f(n)) = \log n$

Summary

Thanks for the Attention!
Questions?

2 Problems, the canonical 2.5-coloring and k-R&C, which is **complete** for the classes $O\left(n^{\frac{1}{k}}\right)$

No Knowledge

- Nodes do not know anything about n
- Unbounded Ids
- K-R&C is $\Omega(n)$ for all k

Randomization

- Helps in a non-trivial way (unexpectedly)
- Quickly gives weird behavior (half-log)

Polynomial Promise

- Nodes are given N,c with promise $n \leq N \leq n^c$
- We can exploit this to get non-trivial improvement
- Solve an optimization problem to get parameter α
- We perfectly capture this tradeoff and get $\Theta(n^{\alpha c})$ for k-R&C

Polynomial Promise on Ids

- Nodes are just c with the promise that Ids $\leq n^c$
- This is just as good for 2.5-coloring, we get $\Theta(n^{\alpha c})$
- This very likely extends to all problems