

# Algorithms for parallel graph exploration

Dominik Pająk

University of Cambridge, UK

LaBRI 3 February 2015

# Exploration of unknown environment

- We enter an **unknown** cave.
- Initially – no knowledge about the cave.



# Exploration of unknown environment

- We enter an **unknown** cave.
- Initially – no knowledge about the cave.
- When we walk in the tunnels we learn about its topology.



# Exploration of unknown environment

- We enter an **unknown** cave.
- Initially – no knowledge about the cave.
- When we walk in the tunnels we learn about its topology.
- We explore more – we learn more.



# Exploration of unknown environment

- We enter an **unknown** cave.
- Initially – no knowledge about the cave.
- When we walk in the tunnels we learn about its topology.
- We explore more – we learn more.
- We want to explore the whole cave.



# Parallel exploration



Can we explore the environment efficiently (quickly) if we have **multiple** walkers (agents) walking **in parallel**?

# Parallel exploration



Can we explore the environment efficiently (quickly) if we have **multiple** walkers (agents) walking **in parallel**?

- We can explore multiple tunnels at the same time.

# Graph exploration

We can model cave as a graph:

- $n$  – number of nodes.
- $m$  – number of edges (tunnels).
- $D$  – distance from the start to the furthest node.

Team of agents

- $k$  – number of agents. All agents start in node  $r$ .
- Agents are moving in parallel.
- Traversal of one edge takes one time step. If we want to model a longer tunnel we can use multiple edges.

Lower bound

- Time  $D$  is needed because some walker has to go to the furthest node.



# Graph exploration

We can model cave as a graph:

- $n$  – number of nodes.
- $m$  – number of edges (tunnels).
- $D$  – distance from the start to the furthest node.

Team of agents

- $k$  – number of agents. All agents start in node  $r$ .
- Agents are moving in parallel.
- Traversal of one edge takes one time step. If we want to model a longer tunnel we can use multiple edges.

Lower bound

- Time  $D$  is needed because some walker has to go to the furthest node.

Goal

We can use many agents but we want to explore in time close to  $D$ .

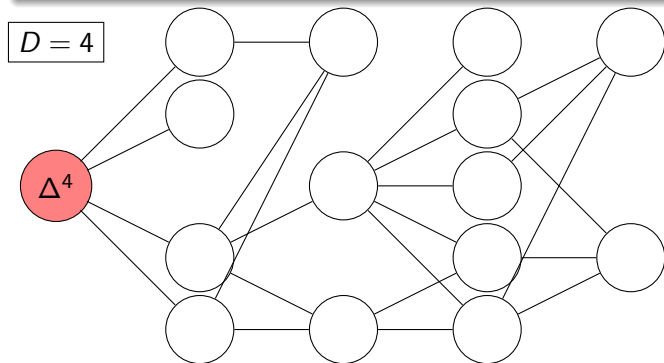
# Naïve approaches

$i$ -th level of the graph – all nodes at distance  $i$  from  $r$ .

## Naïve approach 1: Flooding

Straightforward algorithm to explore in time  $D$  using  $\Theta(\Delta^D)$  agents ( $\Delta$  - maximum degree):

In  $i$ -th step each node at level  $i$  splits evenly all its agents to its neighbours at level  $i + 1$ .



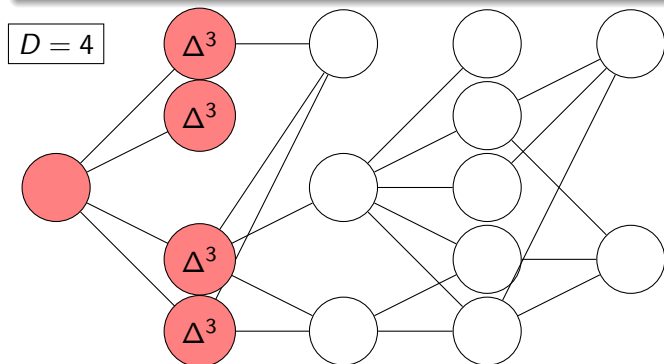
# Naïve approaches

$i$ -th level of the graph – all nodes at distance  $i$  from  $r$ .

## Naïve approach 1: Flooding

Straightforward algorithm to explore in time  $D$  using  $\Theta(\Delta^D)$  agents ( $\Delta$  - maximum degree):

In  $i$ -th step each node at level  $i$  splits evenly all its agents to its neighbours at level  $i + 1$ .



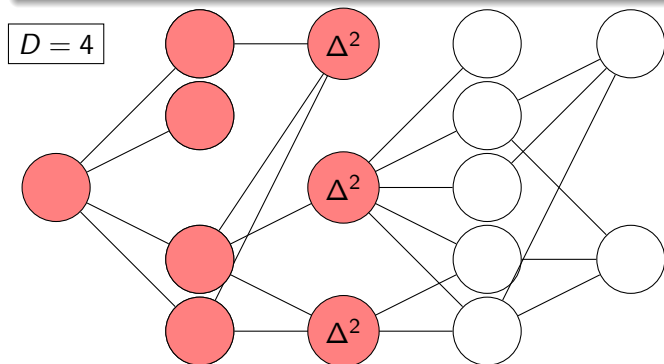
# Naïve approaches

$i$ -th level of the graph – all nodes at distance  $i$  from  $r$ .

## Naïve approach 1: Flooding

Straightforward algorithm to explore in time  $D$  using  $\Theta(\Delta^D)$  agents ( $\Delta$  - maximum degree):

In  $i$ -th step each node at level  $i$  splits evenly all its agents to its neighbours at level  $i + 1$ .



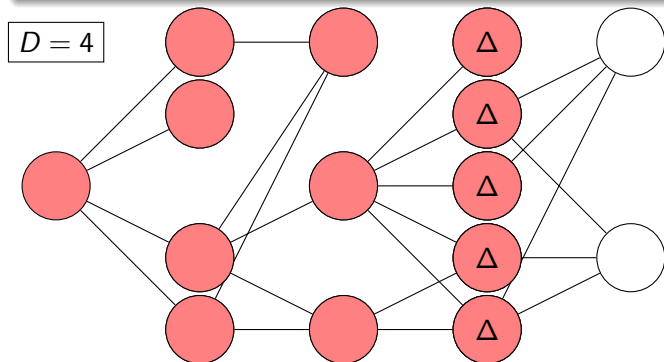
# Naïve approaches

$i$ -th level of the graph – all nodes at distance  $i$  from  $r$ .

## Naïve approach 1: Flooding

Straightforward algorithm to explore in time  $D$  using  $\Theta(\Delta^D)$  agents ( $\Delta$  - maximum degree):

In  $i$ -th step each node at level  $i$  splits evenly all its agents to its neighbours at level  $i + 1$ .



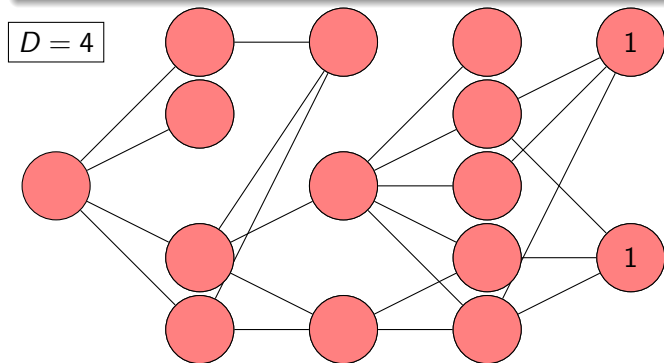
# Naïve approaches

$i$ -th level of the graph – all nodes at distance  $i$  from  $r$ .

## Naïve approach 1: Flooding

Straightforward algorithm to explore in time  $D$  using  $\Theta(\Delta^D)$  agents ( $\Delta$  - maximum degree):

In  $i$ -th step each node at level  $i$  splits evenly all its agents to its neighbours at level  $i + 1$ .



## Unrealistic assumptions

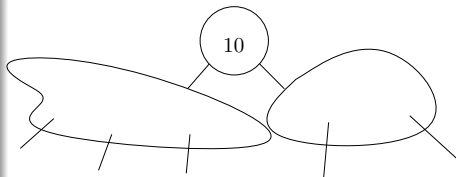
- Assume that the walkers can communicate. Lets say that they have a common knowledge about the progress of exploration.
- Assume that the graph is a tree (no cycles).

## Unrealistic assumptions

- Assume that the walkers can communicate. Lets say that they have a common knowledge about the progress of exploration.
- Assume that the graph is a tree (no cycles).

## Algorithm

- Release the walkers in groups from the root.
- At each node split the agents proportionally to the number of open unexplored tunnels.



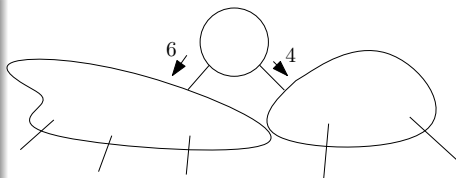


## Unrealistic assumptions

- Assume that the walkers can communicate. Lets say that they have a common knowledge about the progress of exploration.
- Assume that the graph is a tree (no cycles).

## Algorithm

- Release the walkers in groups from the root.
- At each node split the agents proportionally to the number of open unexplored tunnels.



## Theorem

The algorithm explores in time  $O(D)$ , using  $Dn^c$  agents (for any  $c > 1$ ).

- $Dn^c$  is still a lot of agents but at least it is polynomial in  $n$ .

## Theorem

The algorithm explores in time  $O(D)$ , using  $Dn^c$  agents (for any  $c > 1$ ).

- $Dn^c$  is still a lot of agents but at least it is polynomial in  $n$ .
- Both unrealistic assumptions can be dropped:
  - We can explore **any graph** using smart flooding.
  - We can do it using only **local communication**.

## Random walks

- Multiple independent random walks starting from the same position.

## Random walks

- Multiple independent random walks starting from the same position.
- Easy to implement: no memory needed.
- It is still an open problem if any parallelization is guaranteed.
- Poor parallelization on some graphs (e.g., path).

## Random walks

- Multiple independent random walks starting from the same position.
- Easy to implement: no memory needed.
- It is still an open problem if any parallelization is guaranteed.
- Poor parallelization on some graphs (e.g., path).

## Grids

- $O(D^2)$  is possible using polynomial number of agents.
- To get time  $O(D)$ , we need exponential.

## Fair strategies

- Each node is sending (cumulatively over time) the same  $(+/- 1)$  number of agents in every direction.
- Similar to random walks on many graphs.
- $\log k$ -times faster exploration is guaranteed.
- Similarly as for the random walks we need exponential number of agents to explore grids in  $O(D)$ .

- Smart flooding explores any graph in time  $O(D)$  using polynomial number of agents.
- Random walks and fair strategies can also be used for parallel exploration but to explore all graphs in  $O(D)$  exponential number of agents is needed.



Thank You!