Tree Exploration with Logarithmic Memory

- **Speaker:**
  - Xiaohui Zhang

- **Joint work with:**
  - Leszek Gąsieniec, Andrzej Pelc and Tomasz Radzik

- **Special thanks go to:**
  - Dariusz Kowalski, Gad Landau and David Peleg
Graph exploration

- Network (graph and protocols) models:
  - labeled versus anonymous
  - distributed versus centralized
  - synchronized versus asynchronous
  - restricted topologies, e.g., lines, rings, trees, etc

- Robots (mobile agent)
  - oblivious versus adaptive
  - restricted abilities, e.g., limited memory, moves, etc
Trees and robots

Tree model: (anonymous)
- nodes of a tree $T$ have no labels, however
- the endpoints of edges incident to each node have unique labels drawn from the set $\{0,...,d_{\text{max}}-1\}$, where $d_{\text{max}}$ is the maximum degree in the tree

Robot model:
- robot has internal memory limited to $O(k)$ bits
- robot can move across edges at a time chosen according to some predefined deterministic protocol
- robot can be also seen as a deterministic finite automaton with $(2^{O(k)})$ states
Tree - example
The tree exploration task

Assume that a robot has $O(\log n)$ bits of memory and it starts exploration of a tree $T$ in node $s$.

The robot:

1) visits all nodes in $T$ and comes back to $s$, \textit{iff} $T$ is of size $\leq n$, or

2) reports that $T$ is too large otherwise.
Motivation (direct)

- Diks, Fraigniaud, Kranakis and Pelc, SODA’02
- Upper bound $O(\log^2 n)$
- Lower bound $\Omega(\log n)$
- They provided also a lower bound $\Omega(\log\log\log\log n)$ if a robot does not have to go back to the original node on the tree
Euler tour - basic walk \((i \rightarrow i+1)\)

\((i \rightarrow i+1) \mod d\) protocol

Start from the break point: 1201220012001120120012001
Easy ride in trees (basic walk)

- In tree exploration problem the robot is expected to visit *each node* of a tree *at least once*
- In order to visit all nodes the robot can use $i \rightarrow i+1$ protocol (basic walk) for as long as it takes

- How long does it take to visit all nodes?
- How does the robot figure out that the exploration process is accomplished if its memory is limited?
Partial Euler tour

- entire sub-tree
  200112

- sub-tree with open path
  011201220
  02010
  1201220120011201200120012001
Periodicity of Euler tour

- Performing *basic walk*, i.e., going along the *Euler tour* in the tree the labels of placed on edges must form a *periodic sequence* (with the shortest period $\pi = t[1..q]$) eventually.

- This might happen because of:
  - partial repetitive structure of explored tree, or
  - the whole tree has been visited a number of times.

- The main idea of the algorithm is:
  - to look for prefix periodicity in the basic walk and then
  - test whether the period $\pi$ corresponds to the whole tree or not.
**Basic walk periodicity (simple tree)**

- **Case 1:** single occurrence of the period $\pi$ of the basic walk corresponds to the Euler tour of a tree.
Basic walk periodicity (symmetric tree)

**Case 2**: two occurrences of the period $\pi$ of the basic walk corresponds to the Euler tour of a tree.

The period $\pi$ of the basic walk

01220

010

012020 (012020)

Euler tour

01220 01220

Open path is an odd palindrome

01220 01220 01220 (01220)
**Case 3**: neither one nor two occurrences of the period \( \pi \) of the basic walk do not correspond to the Euler tour of a tree.

In this case an iteration along consecutive occurrences of the period \( \pi \) creates an open path, i.e., we do not go back to the starting point in the tree.
Signatures (via cyclic rotations)

- Each outgoing fragment of the Euler tour at a node $v$ corresponds to some cyclic rotation of $\pi$

$\pi$

$(pseudo\text{-})$ size of sub-tree

node $v$ has signature $(i,j,k)$

e.g., $|C|=j-i$
Matching edges

Two distinct edges \((v, w)\) and \((w', v')\) occurring on the basic walk \(\pi = t[1..q]\) match, if:
- edges have the same labels
- nodes \(v\) and \(v'\) have the same signatures, and
- nodes \(w\) and \(w'\) have the same signatures

**Lemma:** Each edge on the basic walk \(\pi = t[1..q]\) has at most one matching edge

(Otherwise \(\pi\) would be periodic!)
Nesting property

Tree should satisfy nesting property, i.e., for every two pairs of indices $1 \leq l < i' \leq q$ and $1 \leq l < j' \leq q$ if edges at positions $i$ and $i'$ match in $\pi$, edges at positions $j$ and $j'$ match in $\pi$, and $i < j$, then either $i < j < j'$ or $i < i' < j < j'$

Pseudo sizes and corresponding fragments on the basic walk $\pi$

This is ok

This is not allowed!
Separation of Cases 1, 2 and 3

On the basic walk $\pi = t[1..q]$ there:

- all edges are matched in Case 1
- exactly one (symmetric edge) is unmatched in Case 2

[In cases 1 and 2 both tests never fail]
- are at least 2 unmatched edges if non of the tests fails.

[The crash of the test also implies that the tree is not explored yet!]

- All test can be performed in the proposed model.
Matched and unmatched edges

Matched edges

Unmatched edge (symmetric edge)
Unmatched edges in Case 3

Basic walk from $r \ t^{[1,20]} : 00112012201201122012$

Unmatched edges in $t^{[1,20]} : (00)(11)(2(01(22)0)(12(0(11)(22)01)2)$
Algorithm Log-Exploration(n);

\[ q := 1; \]

while \( q \leq n \) do

\[ \text{if } (\pi = t[1..q] \text{ is not periodic}) \text{ and } (t[1..3q] = \pi^3) \text{ and } \]
\[ \text{(Node-Signature-OK and Nesting-OK and Unmatched-Edges} \leq 1) \text{ then terminate } \]

else \[ q := q+1; \]

end while
Assume that a robot has $O(\log n)$ internal memory and it starts exploration of a tree $T$ in node $s$.

The robot:

1) visits all nodes in $T$ and comes back to $s$ iff $T$ is of size $\leq n$, or
2) reports that $T$ is too large otherwise.
Open problems

- Can we classify trees according to exploration space complexity? E.g., you can visit a simple path of any length using a robot equipped with constant number of bits.

- What if there are no explicit labels on the endpoints of edges? I.e., when the robot comes to a node a black-box mechanism moves it to the next() port, or next(next()) port, etc.