Undirected Graph Exploration with $\Theta(\log \log n)$ Pebbles

Yann D., Jan Hackfeld, Max Klimm
TU Berlin
Intro
Graph Exploration

- **agent:**
  - oblivious of graph and location
  - all vertices look the same
  - algorithm producing a color sequence  
    → move strategy

- **goal:** visit all vertices

- **graph:**
  - undirected
  - regular ($\Delta = \text{const}$)
  - edge-colored  
    (locally unique)
  - connected
  - starting location $v$
Exhaustive Search

- Is exploration possible, or can we get “trapped”?

- **obs:** some $2(n-1)$-sequence explores the graph

- **algorithm:**
  1. try next sequence
  2. backtrack
  3. repeat

- **issues:**
  - exponential runtime
  - linear memory

- **Can we do better?**
Are \( b \) bits of memory enough?
- e.g. right-hand rule for simple mazes needs \( b = O(1) \)
- in general: \( s = 2^b \) different states

**again**: behaviour is indep. of graph

try infinite 3-regular tree
- after \( 2^b \) steps a state repeats
- close loop
  \[ \rightarrow \text{“trap” of size } 2^b \text{ for one state} \]

agent needs \( b = \Omega(\log n) \) bits

[Fragniaud, Ilcinkas, Peer, Pelc, Peleg; TCS’05]
• **DFS:**
  1. try next sequence
  2. backtrack
  3. repeat

• **random walk:**
  • runtime $O(\Delta^2 n^3 \log n)$ w.h.p
    [Aleliunas, Karp, Lipton, Lovasz, Rackoff; FOCS’79]
  • constant memory

• **sophisticated, deterministic:**
  • DFS on expander
    [Reingold; J.ACM’08]
  • needs $O(\log n)$ memory

• **main issue:**
  • cannot recognise vertices
Pebbles

- **issue:**
  - cannot recognise vertices
  - If we can “mark” vertices?

- **pebbles:**
  - can be dropped, recognised, picked up

- with $\Theta(n)$ pebbles:
  - DFS with constant memory

- with $\Theta(\log n)$ pebbles:
  - simulate $\Theta(\log n)$ memory

- **Can we do with fewer?**
Results

• **question:**
  Amount of pebbles and memory to not get trapped?

• **known bounds** \((p \text{ pebbles, } s=2^b \text{ states})\):
  
  • If \(p = 0\), then \(b = \Theta(\log n)\) bits are necessary&sufficient
    
    [Fragniaud, Ilcinkas, Peer, Pelc, Peleg; TCS’05] & [Reingold; J.ACM’08]
  
  • \(n = O(s^{s^{s^{\ddots^{s}}}})\) \((\Theta(p) \text{ levels in the exponent})\) for exploration
    
    [Rollik; Acta Inf. ’80] & [Fragniaud, Ilcinkas, Rajsbaum, Tixeuil; TCS’06]

• **new results** \([D., Hackfeld, Klimm; SODA’16]\):
  
  • \(b=O((\log n)^{1-\epsilon})\) memory \(\Rightarrow p=\Omega(\log \log n)\) pebbles needed
    
    [more precisely: \(n = O(s^{8p+1})\)]
  
  • algorithm for \(b = p = O(\log \log n)\)
Lower Bound
Barriers

• **recall** ($s$ states, 0 pebbles):
  • $c=s$ possible configurations
  • size $n=c$ trap for state $s_0$
    ⇒ size $n=c^2$ trap for any state

• **issue** (for $p>0$):
  • $c = s \cdot n^p > n$ configurations
    (state & pebble locations)

• **idea:**
  • ensure pebbles stay “close-by”
Recursive Construction

- **0-barrier** $B_0$:
  - $c_0 = s$ configurations, size $|B_0| = s^2$

- **1-barrier** $B_1$:
  - $c_1 > s$ configurations
  - replace edges of $B_0$ with 0-barriers
    $\Rightarrow$ must carry pebble along
  - $B_1$ must work independent of which of the $p$ pebbles we have with us
    - use $p$ copies, one per pebble
    - $|B_1| \approx p \cdot |B_0| \cdot c_1^2$

- **$r$-barrier** $B_r$:
  - recursion $\Rightarrow |B_r| \approx \binom{p}{r} \cdot |B_{r-1}| \cdot c_r^2$

- **Result**: $|B_p| = O(s^{8p+1})$
Exploration Algorithm
Exploration Algorithm

- **recall** (0 pebbles):
  - $\Theta(\log n)$ memory is enough [Reingold; J.ACM’08]
- **idea**:
  - start with $m_0 = O(1)$ memory bits
  - run Reingold’s algorithm
    - discover $\Omega(2^{m_0})$ vertices
  - **Lemma**: get tour $T$ with $\Omega(2^{m_0})$ vertices
  - place $a = O(1)$ pebbles along $T$
    - $|T|^a$ configurations
      $\rightarrow$ encodes $m_1 = \log |T|^a \approx am_0$ memory bits
  - repeat procedure with $m_1$ starting bits
- **after $\log \log n$ steps**:
  - $a^{\log \log n} = \Omega(\log n)$ memory bits $\rightarrow$ enough
  - **Lemma**: steps take $O(1)$ pebbles&memory
- **Result**: expl. with $O(\log \log n)$ pebbles&memory
Summary
If you don’t want to get lost, take

$\Theta(\log n)$ memory

or

$\Theta(\log \log n)$ pebbles (& memory).

Thank you!