Dynamic networks

Model

- *Synchronous* model
- Static vertex set V
- $E : N_0 \rightarrow \binom{V}{2}$, *where* $\binom{V}{2}$ *is the set of all distinct*
- *pairs* $(u, v) \in V \times V$
- So, E(r) is the set of edges in round r
- *1-Interval connectivity*: G_r = (V, E(r)) is connected for every r
- Can be generalized to T-Interval connectivity

Algorithm and adversary

- Deterministic algorithms
- Nodes communicate by anonymous broadcast
  - At the beginning of round r each node decides which message to bcast based on internal state
  - *Independently*, the adversary decides E(r) → the adversary does not know which message the algorithm is about to send
- Nodes start in initial state that contains own IDs and input
- Nodes know nothing about network and initially cannot distinguish it from any other network

Basic problems

- **Counting.** Whenever executed on network with n nodes, all nodes eventually terminate with output n
- **K-verification.** Determine whether or not k <= n
- **K-token dissemination.** Each node u initially receives a set I(u) of tokens from some domain *T*. $| U_u I(u) | = k$. An algorithm solves the problem when all nodes eventually terminate and output $U_u I(u)$
- **K-committee election.** Nodes partition themselves into subsets, called *committees*, such that
  - The size of each committee is at most k
  - If k >= n then there is only one committee

Counting/1

Theorem 7.2. *Assume that there is a single token in the network. Further assume that at time 0 at least one node knows the token and that once they know the token, all nodes broadcast it in every round. In a 1-interval connected graph $G = (V, E)$ with n nodes, after $r \leq n-1$ rounds, at least $r+1$ nodes know the token. Hence, in particular after $n-1$ rounds, all nodes know the token.*

Counting/2

Theorem 7.3. *Counting is impossible in 1-interval connected graphs with asynchronous start.*

- Result also applies to other basic problems as long as we do not assume knowledge of n or upper bound on it

Counting/3

$A \leftarrow \{self\}$;
for $r = 1, 2, \ldots$ **do**
  broadcast A;
  receive $B_1, \ldots, B_s$ from neighbors;
  $A \leftarrow A \cup B_1 \cup \ldots \cup B_s$;
  if $|A| \leq r$ **then** terminate and output $|A|$;
end

Algorithm 1: Counting in linear time using large messages

Counting/4

Lemma 7.4. *Assume that we are given an 1-interval connected graph* $G = (V, E)$ *and that all nodes in* V *execute Algorithm 1. If all nodes together start at time 0, we have* $|A_u(r)| \geq r+1$ *for all* $u \in V$ *and* $r < n$.

Theorem 7.5. *In an 1-interval connected graph* G*, Algorithm 1 terminates at all nodes after* n *rounds and output* n.

Lemma 7.6. *Assume that we are given a 2-interval connected graph* $G = (V, E)$ *and that all nodes in* V *execute Algorithm 1. If node* u *is waken up and starts the algorithm at time* t*, it holds that have* $|A_u(t+2r)| \geq r+1$ *for all* $0 \leq r < n$.

A non distributed solution

Assumption 1.2 (Node Identifiers). *Each node has a unique identifier, e.g., its IP address. We usually assume that each identifier consists of only* $\log n$ *bits if the system has* n *nodes.*

Algorithm 1 Greedy Sequential

while $\exists$ uncolored vertex v **do**
    color v with the minimal color (number) that does not conflict with the already colored neighbors
end while

Performance

Theorem 1.5 (Analysis of Algorithm 1). *The algorithm is correct and terminates in n "steps". The algorithm uses $\Delta + 1$ colors.*

- Δ is the maximum degree of the graph
- The *chromatic number* Ξ(G) of G is the minimum number of colours in a proper vertex coloring of G

A useful subroutine

Procedure 2 First Free

Require: Node Coloring {e.g., node IDs as defined in Assumption 1.2}

Give v the smallest admissible color {i.e., the smallest node color not used by any neighbor}

- Caveat: no two nodes are coloured at the same time

Algorithm for synchronous case

Algorithm 3 Reduce

Assume that initially all nodes have ID's (Assumption 1.2)
Each node v executes the following code
node v sends its ID to all neighbors
node v receives IDs of neighbors
while node v has an uncolored neighbor with higher ID **do**
    node v sends "undecided" to all neighbors
    node v receives new decisions from neighbors
end while
node v chooses a free color using subroutine **First Free** (Procedure 2)
node v informs all its neighbors about its choice

- Thm.: algorithm is correct and has time complexity n. It uses $\Delta + 1$ colours

Trees

Lemma 1.9. $\chi(Tree) \leq 2$

Algorithm 4 Slow Tree Coloring

```
Color the root 0, root sends 0 to its children
Each node v concurrently executes the following code:
if node v receives a message x (from parent) then
    node v chooses color c_v = 1 − x
    node v sends c_v to its children (all neighbors except parent)
end if
```

- Caveat: how do we choose a root?

Trees/cont.

- The previous algorithm also works in the asynchronous case
  - Time complexity is the tree height
  - Message complexity is n-1
- Is it possible to do better?
- log*n time is possible!