Local Distributed Computing

Porquerolles, France, May 15-19, 2017

This problem is about solving *weak coloring* of connected networks, that is, the objective is to assign a color to each node such that every node has at least one neighbor with a different color.

Question 1. Show that every connected network with at least two nodes is weakly 2-colorable (i.e., every connected network with $n \geq 2$ nodes has a weak coloring using just two colors).

Question 2. Propose a randomized distributed algorithm for weak 2-coloring in the LOCAL model, terminating in $O(\log n)$ rounds with probability at least $1 - O(1/n)$ in n-node networks. (Prove the correctness of your algorithm, and explain in details the arguments used in its analysis).

Question 3. Let G be a weakly c-colored graph, with $c > 4$. Let c' be the smallest integer such that $\binom{c'}{\lfloor c'/2 \rfloor} \geq c$. Using the subsets of $\{1, \ldots, c'\}$ of cardinality $\lfloor c'/2 \rfloor$, show how to reduce the number of colors from c to c' in one round.

Question 4. Assume that G is weakly c-colored with $2 < c \leq 4$. Show how to reduce the number of colors to 2 in a constant number of rounds.

Question 5. Show that, starting from a graph G that is weakly k-colored for some $k = O(1)$, one can weakly 2-color G in a constant number of rounds. (One may use the fact $\binom{x}{\lfloor x/2 \rfloor} = \frac{4^x}{\sqrt{\pi x}}(1 + o(1))$).

— The End —

EPIT2017: Lower Bounds in Distributed Computing Exercises

1 Covering and valence in consensus algorithms

Consider an obstruction-free binary consensus algorithm using atomic read-write registers. Recall that a configuration of such an algorithm specifies local states of the processes and states of the shared registers.

Given a set of processes $P \subseteq \Pi$ and a configuration C, we say that P is *v-valent from C* if v is the only value that can be decided in $C\alpha$ for any P-only schedule α. Respectively, P is *bivalent from C* if it is not v-valent for any $v \in \{0, 1\}$.

Let P be bivalent from $C\beta$, where β is a block write by some $R \subseteq P$. Let γ be a schedule of some $z \notin P$ such that z decides in $C\gamma$. Show that z must write to a register not covered by R in C.

2 Space complexity of mutual exclusion

Recall that in the *mutual exclusion* problem, every process is in the *remainder section* (RS) initially, then it may enter its *trying section* (TS), then to its *critical section* (CS), and then to its *exit section* (ES), after which it returns to its remaineer section.

It is required that:

- no two processes are in their critical sections at the same time, and
- assuming that no process fails in its trying or exit sections and never stays in its critical section forever, at least one process in its trying section will eventually enter its critical section.

1. Show that any 2-process read-write mutual-exclusion alogirithm requires at least 2 registers.
2. What about 3 processes? Can you show that 3 registers are necessary?
3. Finally, prove the general statement: n-process algorithm requires n registers.

   Hint: starting with an execution E in which a single process enters its critical section, use other processes, one by one, to "perturb" E by covering distinct registers in such a way that no process can "witness" that they are not in the remainder section. It is recommended to "reuse" processes by forcing them to enter CS over and over again (passing through ES, RS, and TS).

Prof. Patt-Shamir

Exercise

1. Consider the problem of finding the maximum input element, where inputs are from the domain $[1, M]$ for some natural $M > 1$. Design a deterministic protocol that, given $\varepsilon > 0$, finds the maximum to within a $(1 + \varepsilon)$ factor in time D and uses messages of size $O(\varepsilon^{-1} \log \log M)$.
   Hint: Use the fact that $\log_{1+\varepsilon} x = \Theta(\varepsilon^{-1} \log x)$ for $0 < \varepsilon \leq 1$.

2. Prove that the number of nodes n cannot be approximated to within any finite factor f by a uniform (in particular, no IDs) deterministic protocol.
   Hint: Think of two systems, one with n nodes and the other with $(f + 1)n$ nodes in similarly looking topologies, and analyze the evolution of a symmetric schedule.

3. Consider the Bellman-Ford algorithm in a synchronous environment with a single source node s. In each round, the source sets $d_s := 0$, and any other node v sets $d_v := \min\{d_u + 1 \mid (u, v) \in E\}$. Assume that the variable d can take only non-negative values. Prove that regardless of the initial state of the d variables, for any node v, d_v eventually stabilizes on $\text{dist}(s, v)$.
   Hint: Prove that $d_v \leq \text{dist}(s, v)$ after $\text{dist}(s, v)$ rounds by induction on time, and similarly that $d_v \geq \text{dist}(s, v)$.
   Extra credit: Prove the same claim for weighted graphs. What is the stabilization time in this case?

4. Consider the MIS algorithm and analysis. Prove (by an argument) or disprove (by an example): an edge can be removed but not killed.

5. Prove that with probability $1 - n^{-\Omega(1)}$, the MIS algorithm eliminates all edges in $O(\log n)$ rounds.
   Hint: Call a round is successful if it eliminates at least a $1/4$ of the edges. Use Markov's inequality to prove that every round is successful with constant probability. Then use Chernoff's bound to show that w.h.p., out of any $O(\log n)$ rounds of the MIS algorithm, $\Omega(\log n)$ rounds are successful.

6. Prove that if k messages are routed along paths which are at most one edge longer than their source-destination distance, then the number of delays cannot be greater than $2k - 2$. Give a parametric example (in k) where paths are at most one hop linger than the shortest, and there is a message which actually suffers $2k - 2$ delays. What's the *average* delay in you example?

Spring School on Theoretical Informatics

Porquerolles, May 14-19, 2017

Simple exercises related to Michel Raynal's lectures

1 Exercise 1: on the kFK universal construction

Slides 44-56 presented a very simple universal construction suited to the computation model $\mathcal{CARW}$[LL/SC]. This construction uses an internal procedure apply(), which is based on a "repeat twice" loop statement. Let us redefine this operation where "repeat twice" is replaced by a "repeat until" statement as follows:

```
internal procedure apply() is
  repeat
     ls ← STATE.LL();
     pairs ← BOARD.collect();
     for ℓ ∈ {1, 2, ··· , n} do
          if (pairs[ℓ].sn = ls.sn[ℓ] + 1 then
               ⟨new_state, r⟩ ← δ(ls.value, pairs[ℓ].op);
               ls.res[ℓ] ← r; ls.sn[ℓ] ← pairs[ℓ].sn
          end if
     end for
  until STATE.SC(ls) end repeat.
```

Is this modification correct? If it is correct provide a proof of it. If it is incorrect, provide a counter-example.

2 Exercise 2: on operations on memory locations

Let us consider the model $\mathcal{CARW}[\emptyset]$ enriched with the following atomic hardware-provided operations. Hence (as the read and write operations) these operations can access any memory location, or the very same location. Let X denote a memory location, and α an integer greater than 1.

- X.multiply(α) multiplies by α the value in X. (Hence if $X = x$ when X.multiply(α) is invoked we have $X = \alpha \times x$ when it returns.)
- X.decrement() decrements by 1 the value in X. (Hence if $X = x$ when X.decrement() is invoked we have $X = x - 1$ when it returns.)

Show that in the system $\mathcal{CARW}[\emptyset]$ enriched with multiply(α) and decrement() (in addition to read()), consensus can be solved for ANY number of processes. To this end design (and prove correct) in this computing model, a binary consensus algorithm which works for any number of processes.

If time permits (... much more difficult ...) try to show that binary consensus for 2 processes is impossible in $\mathcal{CARW}[\emptyset]$ enriched with only one of the operations decrement() or multiply(α)). (Actually, $\mathcal{CARW}[\emptyset]$ enriched with only one of these operations has consensus number 1.)

EPIT 2017 - Mobile Networks - Exercices

Sébastien Tixeuil

May 17, 2017

1 Stateless Flooding

The network is represented as a graph $G=(V,E)$, where V is a set of devices, and E is a set of edges that connect them. An edge exists between two devices if they can send messages directly. Two such devices are called *neighbors*. The graph is *fixed maximum degree* if there is constant k, independent of network parameters, such that each device has at most k neighbors. The communication is bi-directional and the graph is undirected. A network is *connected* if there exists a path between any two devices. Two messages are *mates* if the sender of each message is the receiver of the other.

Every device has a *send queue* SQ that collects messages to be sent. A message is transmitted by taking it from the sender's send queue, transferring it to the receiver and processing it according to the routing algorithm. In this paper, we assume that this transferal and processing is done in a single atomic *step*. The *atomicity* of the step means that it may not overlap with steps on this or other devices. In practice, only the neighbor device steps may not overlap.

Computation is a sequence of atomic steps that starts in an initial state of the algorithm. A computation is *fair* if every message that is in a send queue SQ of some device is eventually either transmitted or removed from this queue during this computation. That is, a message may not "get stuck" in a send queue forever. To reason about a routing algorithm, we consider its fair computations. A computation is *finite* if it has a finite number of steps. A routing algorithm is *terminating* if all its computations are finite. A terminating routing algorithm never leaves messages indefinitely circulating in the network.

The pseudcode for a tentative *stateless flooding* (SF) routing algorithm is shown in Figure 1. The algorithm is as follows. The source device adds a message $M(sender,receiver)$ to its send queue SQ to be sent to all devices in its neighbor set N. When a device receives a message from neighbor a, it first checks its send queue for a mate. If a mate exists, both messages are discarded. Otherwise, the device sends the message to all neighbors except a.

```
device s
    foreach n∈N do
        add M(s,n) to SQ

device n
if receive M(a,n) then
    if M(n,a)∈SQ then
        /* found mate */
        discard M(n,a) from SQ
    else
        foreach m∈N:m≠a do
            add M(n,m) to SQ
```

Figure 1: SF pseudocode.

Exercise 1. Show that SF guarantees termination and delivery from the source to all target devices connected to the source.