CMPSCI 711: More Advanced Algorithms

Section 6-2: Distributed Functional Monitoring

Andrew McGregor

Last Compiled: April 29, 2012

Distributed Functional Monitoring

- k streams $S_1, S_2, \ldots, S_k$ are being observed at k locations.
- Messages are sent between the observers and a central site.
- How many messages need to be sent such that central site always knows $f(S_1 \cup \ldots \cup S_k)$, e.g., whether the total number of elements have passed some threshold τ.
- *Contrast with Communication Complexity:* Need to track value of f dynamically. Measure number of messages rather than total size.

Outline

Counting

Distributed Sampling

Counting

- *Goal:* Determine when the total count passes threshold τ.
- First attempt:
  - Player i maintains $n_i =$ number of elements seen locally since last update to central site.
  - Player i sends n_i to central site if it reaches τ/k
  - On receiving n from some player, central site sets $\tau \leftarrow \tau - n$ and broadcasts new threshold to call players.
- Correctness is clear since $\sum_i n_i < k \cdot \tau/k < \tau$.
- *Thm:* Total number of messages is $O(k^2 \log(\tau/k))$.

First Attempt Analysis

- Let $\tau_0 = \tau, \tau_1, \tau_2, \ldots$ be the sequence of thresholds.
- Note that $\tau_i \leq (1 - 1/k)\tau_{i-1}$ and hence

$$\tau_r \leq \tau/(1 - 1/k)^r$$

- Therefore after $r = \log_{1-1/k} \tau/k$ rounds $\tau_r \leq k$.
- Each rounds requires $O(k)$ messages so number of messages so far is

$$O(k \log_{1-1/k} \tau/k) = O(k^2 \log \tau/k)$$

- There are at most k more messages so total is

$$k + O(k^2 \log \tau/k) = O(k^2 \log \tau/k) \ .$$

Better Counting Algorithm

- Better algorithm: Proceeds in rounds $j = 1, 2, 3 \ldots$
  - Player i maintains n_i as before.
  - In round j, player sends n_i if it reaches $\tau/(2^j k)$
  - Central site waits until k messages are received then broadcasts the start of the new round.
- Correctness is clear since during round j, $\sum_i n_i < k \cdot \tau/(2^j k) < \tau/2^j$ and total count registered by central site at start of round is

$$\tau/2 + \tau/4 + \ldots + \tau/2^{j-1} = \tau - \tau/2^{j-1} \ .$$

- *Thm:* Total number of messages is $O(k \log(\tau/k))$.

Second Attempt Analysis

- Number of rounds is $\log(\tau/k)$ since $j > \log(\tau/k)$ implies
$$\tau/(2^j k) < 1 \ .$$
- Each rounds has $O(k)$ messages, so total number of messages is
$$O(k \log(\tau/k)) \ .$$

Outline

Counting

Distributed Sampling

Distributed Sampling

- Suppose we want to maintain a random set of s elements from

$$S_1 \cup S_2 \cup \ldots \cup S_k$$

- *Basic Idea:* When each element x is observed by a player, it is assigned a random weight $w(x) \in_R [0, 1]$.The s elements from $S_1 \cup S_2 \cup \ldots \cup S_k$ with the smallest weights are a random subset.

Sampling Algorithm

- *Stored Values:* Central site maintains a set S of $\leq s$ elements and

$$u = \max\{w(x) : x \in S\} \ .$$

  and each player i maintains some $u_i \geq u$
- *Player:* On seeing x, player sends $(x, w(x))$ if $w(x) < u_i$.
- *Central Site:* Updates S such that it includes the s elements from $S \cup \{x\}$ with the smallest weights. Sends u to player i.
- *Central Site:* Broadcasts u if u has decreased by a factor r since last broadcast. Call this a "new round."

Sketch of Analysis

- Let R be the number of rounds. Then

$$\mathbb{E}[R] \leq O(\log_r(n/s))$$

  since final value of u should be about s/n and u decreases by $1/r$ with each round.
- Expected number of messages per round is about $O(rs + k)$.
  - Every sent x gets added to S with probability at least $1/r$.
  - Every update to S decreases u by around a $(1 - 1/s)$ factor.
  - Hence expect $O(rs)$ messages before the broadcast in a round.
- Total messages is $O((rs + k)\log_r(n/s))$ and optimize r to get:

$$\begin{aligned} O(k\log(n/s)/\log(k/s)) \qquad & \text{if } s < k/8 \\ O(s\log(n/s)) \qquad & \text{if } s \geq k/8 \end{aligned}$$