Partially Dynamic Concurrent Update of Distributed Shortest Paths

Serafino Cicerone
Gabriele Di Stefano

Gianlorenzo D’Angelo
Daniele Frigioni

Dept. of Electrical and Information Engineering - Univ. of L’Aquila - Italy
{cicerone,gdangelo,gabriele,frigioni}@ing.univaq.it

Work partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL)
Purpose

Update shortest paths in a graph representing a distributed asynchronous system (e.g. the Internet) when edge changes occur.

- Admitted edge changes:
  - weight increase/delete
  - weight decrease/insert

- The changes can occur in an unpredictable way (concurrent updates)
Purpose

Sequential edge delete

Concurrent edge delete
Outline

- Previous works
- Results of the paper
- Complexity measures
- Decremental algorithm
- Incremental algorithm
- Conclusions and future works
Previous Works

- Classical Bellman-Ford and its variations:
  - Complexity: exponential
  - Drawback: Counting Infinity and Looping phenomena
- Awerbuch, Cidon and Kutten, 1990
  - Complexity: $\Theta(n)$ messages and $O(n^2)$ space per node
  - Drawback: is not able to concurrently update shortest paths
- Italiano, 1991
  - Complexity: $O(n \log(nW))$ messages and $O(n)$ space per node
  - Drawback: is not able to concurrently update shortest paths
- Ramarao and Venkatesan, 1992
  - Complexity: $O(n^3)$ messages and $O(n)$ space per node
  - Drawback: is not able to concurrently update shortest paths
- Cicerone et al, 2003
  - Complexity: $O(\text{maxdeg } \Delta_\sigma)$ messages and $O(n)$ space per node
  - Drawback: is not able to concurrently update shortest paths
Results of the paper

By the above analysis, two classes of algorithms are known:

2. Those which are not able to concurrently update shortest paths
3. Those which are able to concurrently update shortest paths but
   - either they suffer of the looping and counting phenomena, or
   - their convergence can be very slow in the case of weight increase operations (possibly infinite)

This paper provides a partially dynamic algorithm that:
- is able to concurrently update shortest paths
- avoids the looping and counting phenomena
- converges fast
Results of the paper

In Detail:

- we propose a new decremental algorithm
  - Complexity: $O(\text{maxdeg } \Delta^2)$ messages and $O(\text{maxdeg } \Delta)$ space per node
  - is able to concurrently update shortest paths

- we propose a new incremental algorithm
  - works also in the concurrent case
  - Complexity: $O(\text{maxdeg } \Delta)$ messages and $O(n)$ space per node

Here $\Delta$ is the number of nodes affected by a set of weight change operations
Complexity measures

Given a set of $k$ weight changes $\sigma_1, \sigma_2, \ldots, \sigma_k$ and a source node $s$:

- $\delta_{\sigma_i,s}$ : set of nodes that change the shortest paths toward $s$ as a consequence of $\sigma_i$

- $\bigcup_{s \in V} \delta_{\sigma_i,s}$ : nodes affected by $\sigma_i$

- the number of affected nodes is at most:

$$\Delta = \sum_{i=1}^{k} \sum_{s \in V} |\delta_{\sigma_i,s}|$$

We give the complexity bounds as a function of $\Delta$
Decremental Algorithm - data structures

- There are 2 shortest paths:
  - $v$–$s$ distance, $d(v,s)$: weight of the shortest paths (10)
  - $v$–$s$ via, $via(v,s)$: subset of $N(v)$ containing nodes in a shortest path

- Data structures:
  - $d[v,s]$: estimated distance from $v$ to $s$
  - $via[v,s]$: estimated via from $v$ to $s$
Decremental Algorithm - definitions

white: \( d(v,s) = d^k(v,s) \)
\[\text{via}(v,s) = \text{via}^k(v,s)\]

gray: \( d(v,s) = d^k(v,s) \)
\[\text{via}(v,s) \neq \text{via}^k(v,s)\]

black: \( d(v,s) \neq d^k(v,s) \)
Decremental Algorithm - behavior
Decremental Algorithm – complexity

- Message Complexity
  - Only black nodes send messages
  - For each source $s$, each black node $v$ sends $\text{deg}(v)$ messages at each update
  - There are at most $|\delta_{\pi,i,s}|$ updates
  - There are $|\delta_{\pi,i,s}|$ black nodes

$$\sum_{i=1}^{k} \sum_{s \in V} (\text{maxdeg} \cdot |\delta_{\pi,i,s}|^2) \leq \text{maxdeg} \cdot \Delta^2$$

- Space Complexity
  - $\text{via}[v,s]$ contains at most $\text{deg}(v)$ elements
  - $\text{via}[v,*] \text{ requires } O(\text{deg}(v) \cdot n)$
  - $d[v,*] \text{ requires } O(n)$
  - At most: $O(\text{maxdeg} \cdot n)$
Incremental Algorithm - data structures

The same as the decremental algorithm but:

\[ \text{via}[v,s] \text{ stores only one node in } \text{via}(v,s) \]

\[ \text{via}[*,s] \text{ induces a shortest paths tree} \]
Incremental Algorithm - behavior

- $y$ is closest than $x$ to $s$, then $y$ updates its data structures.
- Each shortest path that changes contains the edge $(x, y)$.
- $y$ propagates the algorithm in the shortest paths tree induced by via $[* , y]$. 
Incremental Algorithm – complexity

- **Message Complexity**
  - For each source $s$, for each node $v$, there are at most $|\delta_{\sigma_i,s}|$ updates
  - $v$ sends $deg(v)$ messages at each update

  $$\sum_{i=1}^{k} \sum_{s \in V} (maxdeg \cdot |\delta_{\sigma_i,s}|) = maxdeg \cdot \Delta$$

- **Space Complexity**
  - $via[v,\ast]$ requires $O(n)$
  - $d[v,\ast]$ requires $O(n)$
Conclusions and future works

- We propose a pair of partially dynamic algorithms that are able to concurrently update shortest paths
  - Decremental algorithm complexity: $O(\text{maxdeg } \Delta^2)$ messages and $O(\text{maxdeg } \Delta)$ space per node
  - Incremental algorithm complexity: $O(\text{maxdeg } \Delta)$ messages and $O(n)$ space per node

- Future works:
  - Fully dynamic algorithms
  - Experimental evaluation