Dynamic Algorithms
via
Forbidden-Set Labeling

Cyril Gavoille
(LaBRI, University of Bordeaux)
Contents

1. Generalities on dynamic algorithms
2. Forbidden-set data-structures
3. Forbidden-set routing schemes
Contents

1. Generalities on dynamic algorithms
2. Forbidden-set data-structures
3. Forbidden-set routing schemes
Queries in Dynamic Graphs

Maintaining **data-structures** for dynamic graphs (node/edge addition/deletion) supporting queries like:

- Connectivity: $\text{Find}_G(u)$?
- Approximate distances: $d_G(s,t)$?
- Near-shortest path routing: $\text{Next-Hop}_G(s,t)$?
- ...

Query $(s,t)$, Update, Query$(s',t')$, Update, ...
Goals for a Dynamic Scenario

- Fast **query** time (must be $<<$ time to answer the query in $G$ without pre-processing)

  Ex: $d_G(s,t)$. Instead of $O(m+n\log n)$ time for Dijkstra, prefer $O(n^{\varepsilon})$ or even $\text{polylog}(n)$ query time

- Fast **update** time (must be $<<$ pre-processing time)

  Ex: instead of $O(n^3)$ time for an All-Shortest-Path-Pair algorithm, prefer $O(n^{\varepsilon})$ or $\text{polylog}(n)$ update time
Observation

**Fast** update time ➔ **Small** data-structure

If the space is $S(n)$, then amortized update time must be $\geq S(n)/n$ (starting for $G=\emptyset$ and adding $n$ nodes).

A dynamic scenario with low update time requires a “compact” data-structure solution in the static scenario.

Here compact does not mean to store with **gzip**, but to only store what you need (possibly in a clever way).
An Algorithmic Challenge

♦ Optimal solutions exist ... for trees [Tarjan, Cole, ...]
♦ Connectivity is still open for dynamic general graphs
♦ Widely open for distance & routing queries

In general, node deletion is the most costly operation. In this talk focus on scenario:

Q(s, t), Delete x, Q(s', t), Delete x', ...
Contents

1. Generalities on dynamic algorithms
2. Forbidden-set data-structures
3. Forbidden-set routing schemes
Forbidden-Set Queries

Consider available a data-structure supporting query $Q^*(s,t,X)$, the query $Q(s,t)$ in the graph $G\setminus X$, for a static graph $G$ and for any $\{s,t\} \cup X \subseteq V(G)$

We can solve the dynamic scenario using $Q^*$ as follows:

- If node $x$ is deleted, then just update the set $X$ of forbidden nodes
- If ask for $Q(s,t)$ in $G\setminus X$, then use $Q^*(s,t,X)$ on $G$
- If query time is too big, then recompute a static data-structure for $Q^*$ for the new static graph $G' = G \setminus X$
Low Amortized Updates

Assume $G$ is a sparse graph, has $n$ nodes, and:

♦ Pre-processing time for $Q^*$ in $G$ is $\ldots \ n\log n$
♦ Query time of $Q^*(s,t,X)$ is $\ldots \ |X|\cdot \log n$

Then, recompute $Q^*$ whenever $|X| \geq \sqrt{2n}$

♦ Query time is $\leq \sqrt{2n}\cdot \log n$
♦ Amortized update time for $n$ operations:

$$\frac{(1+2+\ldots +|X|)\cdot \log n + n\log n)}{|X|} \approx \sqrt{2n}\cdot \log n$$

Sublinear!
Contents

1. Generalities on dynamic algorithms
2. Forbidden-set data-structures
3. Forbidden-set routing schemes
The Compact Routing Problem

Input: a network $G$ (a connected graph)
Output: a routing scheme for $G$

A routing scheme allows any source node to route messages to any destination node, given the destination’s network identifier.

Node identifiers can be chosen by the designer of the scheme as a routing label whose length is a parameter.
Routes are constructed in a distributed manner ... according to some local routing tables (or routing algorithms).
... and subgraphs of the grid?

(x,y)-coordinates no longer sufficient; routing in planar graphs...

Routes are constructed in a distributed manner... according to some local routing tables (or routing algorithms)
**Quality & Complexity Measures**

- Near-shortest paths: $|\text{route}(s,t)| \leq \text{stretch} \cdot d_G(s,t)$
- Size of the labels and routing tables
- **Goal:** constant stretch & compact (polylog) tables/labels

Trivial upper bound: $O(n \log n)$ bits, each node stores the neighbour on the next-hop towards each destination
Routing in Static Planar Graphs

Stretch-1 [G. et al., J. Algorithms’ 04]

Shortest-path routing on weighted planar graphs requires labels of \( \Omega(\sqrt{n}) \) bits. Treewidth-k graphs have stretch-1 routing schemes with \( O(k \log^2 n) \)-bit labels. For planar, \( k=\sqrt{n} \).

Stretch \( > 1 \) [Thorup, JACM’ 04]

Weighted planar graphs have \( (1+\varepsilon) \)-stretch routing schemes with \( (1/\varepsilon) \cdot O(\log^2 n) \)-bit labels.
Forbidden-Set Routing

Shortest path avoiding forbidden blue nodes

(2,3)

(5,8)
Forbidden-Set Routing

Input: a network $G$

Output: a forbidden-set routing scheme for $G$

A forbidden-set routing scheme allows any source node $s$ to route messages to any destination node $t$, avoiding any set $X$ of forbidden nodes or edges, given the local table of $s$, the identifier of $t$ and the identifiers of nodes/edges in $X$. 
Motivations

Routing around failures

- Routing schemes are generally static; recomputation of labels/routing tables is costly.
- The set $X$ can be a set of failed nodes/edges
- Best known techniques only handle single failures e.g. “fast reroute”, Cisco not-via

Internet routing

- ASes want control over where their packets travel; shortest-path routing not expressive enough
- BGP allows AS $i$ to specify that its packets avoid AS $j$
Forbidden-Set Routing

[Upper bounds]
O(nlogn) no longer trivial!

The trivial upper bound is to store the entire graph at each node \( \Rightarrow O(n^2) \) bits/node.

[Lower bounds]
Static scenario applies (take \( X=\emptyset \)), i.e., \( \Omega(n) \) for general graphs, and \( \Omega(\sqrt{n}) \) for planar.
Known Results on Forbidden-Set Labeling

[Courcelle-Twigg, STACS’ 07]
Stretch-1 forbidden-set distance and routing in treewidth-k graphs with $O(k^2 \log^2 n)$-bit labels.

[Chechik-G.-Peleg, PODC’ 10]
Stretch-$(1+\varepsilon)$ forbidden-set distance and routing in unweighted graphs of doubling dimension with labels of $(1/\varepsilon) \cdot \text{polylog}(n)$-bit labels.
New Results

[Abraham-Chechik-G., 2011]

Stretch-(1+ε) forbidden-set distance and routing in weighted planar graphs with \((1/\varepsilon) \cdot \text{polylog}(n)\)-bit labels.

\[ \Rightarrow \text{Corollary: } \sqrt{n} \cdot \text{polylog}(n) \text{ worst-case query and update time for } (1+\varepsilon)\text{-approximated distance oracle in dynamic planar graphs.} \]

Previous bound: \(n^{2/3}\) [Klein et al., Algorithmica’98]
Conclusion

A proof of concept:

Forbidden-Set Labeling Schemes with short labels (i.e., local & compact data-structures) indeed do help for the design of efficient data-structures in dynamic graphs.

Thank you!