A journey through compact routing

Ittai Abraham, VMware Research

My personal journey: started around a distributed graph algorithm problem and grown into many other directions

1. Compact routing: a fundamental distributed graph algorithm problem
   - Thanks to: Dahlia, Cyril
2. General techniques: decompose into trees, sampling, partitioning
   - Thanks to: Yair, Ofer
3. Forbidden-set routing: from monadic second order logic through compact routing to dynamic graph algorithms
   - Thanks to: David, Shiri, Sebastian
4. Algorithm engineering: from compact routing on doubling graphs to hub labels and Bing maps
   - Thanks to: Andrew, Renato, Daniel
5. New directions: from compact routing to compact cutting. New connections between cuts, distances and spectral graph theory
   - Thanks to: Richard, Yiannis, David

“The important ideas of combinatorics do not usually appear in the form of precisely stated theorems, but more often as general principles of wide applicability”

The Two Cultures of Mathematics. Gowers

Three Messages

- (Try to) solve hard problems
- (Seek to) understand general principals
- Decompose graph problems into trees

Compact Routing: a fundamental distributed graph algorithm problem

- A distributed network of nodes *G=(V,E)*
- Sender *s* wants to send a message to target *t*
- Goal: route packet from *s* to *t*
- Each node has a routing table
  - When a message arrives to node *u*: a routing scheme looks at message header and routing table to decide how to forward message
  - The decision is a name of an outgoing link

Compact Routing: a fundamental distributed graph algorithm problem

- General graph: each node stores a forwarding table for each destination
  - Routing on shortest path, can be done in *O(n log n)* bits
- Grid graph: each node stores its location *(x,y)* on the grid
  - Routing on shortest path, can be done in *O(log n)* bits

- Fundamental tradeoff: different graph families have
  - Different solutions
  - Different bounds
    - beware of lower bounds on obscure graphs
  - Finding right tradeoff often very important when solving real world problems

Compact Routing 101

- The classic tradeoff: space vs stretch
  - *Space*: max size of routing table
  - *Stretch*: |*route(x,y)*|<= *stretch dist(x,y)*
- Two variants:
  - Labeled: destinations are given a short poly-log sized name
  - Name independent: destinations have arbitrary names (and can even move)

- Key result: for any parameter *k*, can provide each node with a short (poly-log) label
  - Each routing table is of size $O(n^{1/k}\ poly\text{-}log)$
  - Stretch is *4k-5* (Or *O(k)* if nodes use their own names)
- Open: can you get stretch *2k-1* (with size $O(n^{1/k}\ poly\text{-}log)$ table size) ?

Labeled routing on a tree

- Fix a rooted spanning tree *T* in *G*
- Goal:
  - Provide a *label* for each node
  - Given label of *u*, route form *root* to *u* on *T*
- Trivial labeling:
  - Label of node u: the name of the edges from the root to u
  - Stretch is 1, but size of label may be large *O(n)*
- Define: tree edge *(x,y)* is heavy: *y's* subtree contains more than *half* the nodes of *x's* subtree
  - Each node has *at most one heavy* edge towards its children
- Label of node *u*: the *non-heavy edges* from the *root* to *u*

- Stretch is 1, label size is poly-log, routing table is poly-log

$O(n^{1/2}$ poly-log) storage and low stretch

- Each node stores routing table for the $O(n^{1/2})$ closest nodes (denote $B(u)$)
  - So if t belongs to $B(s)$ then we can route to t with *stretch 1*
- Uniformly randomly choose $O(n^{1/2} \log n)$ special nodes (call this set L)
  - So each node u has in expectation $O(\log n)$ nodes in $B(u) \cap L$
- Build a shortest path spanning tree T_l for each special node in L
- Label of t contains:
  - l_t, a special node in $B(t)$
  - $L(t,l)$ the tree routing label of t in tree T_l
- If t not in $B(s)$ then look at t's label and route to l_t and using $L(t,l)$ to t
- Storage for u: $B(u)$, tree routing from each T_l
- Stretch: 5 (can get 3, even name independant, but need more ideas)

Compact Routing: Techniques with wide applicability

- Decompose problem into trees
- Random sampling/ Hitting sets
- Bounded diameter partitions

Bounded diameter partitions

- Fix two parameters $C<D$
- Consider a set Z of of balls of radius C
- Build a partition of clusters, each cluster has bounded diameter D
- Goal: cut as few of the balls of Z as possible

1. Deterministic ball growing (find sparse cut)
2. Randomized ball growing (randomly choose exponential radius)
3. Randomized center choosing on a metric (choose centers via random permutation)
4. Parallel region growing (choose random start time)

From bounded diameter partitions to compact routing

- Build poly-log partitions so that every ball C of radius 2^i has some cluster X in some partition that contains C and X has diameter at most $O(\log n\, 2^i)$
- Build this for each scale
- Build a routing tree on each cluster
- Label for u: the tree label on each scale for the cluster that contains the 2^i ball around u
- Assume $2^{i-1}<dist(s,t)<2^i$, route from s to *root* of scale i cluster mentioned in t's label then from *root* to t

- Overall storage (routing table) is *poly-log*
- Overall stretch is $O(\log n)$

A random bounded diameter partition

- Build *O(log n)* partitions so that every ball *C* of radius 2^i has some cluster *X* in some partition that contains *C,* and *X* has diameter at most $O(\log n\, 2^i)$
- Choose an uncovered *center x:*
  - *Grow* 2^i
  - With *probability ½ stop, otherwise repeat*
- Diameter is bounded by $O(\log n\, 2^i)$ whp
- Fix a ball *C* of radius 2^i
- Consider the first time a cluster reaches *C*, with probability ¼, *C* will be contained in cluster
- Build *O(log n)* random partitions so whp, every *C* has a cluster that contains it

Graph family tradeoffs

- Directed graphs
- Graphs with bounded doubling dimension
  - Bounded highway dimension
- Graphs excluding a fixed minor
  - Planar graphs
  - Bounded genus graphs
  - Bounded treewidth graphs

- Less interesting: use standard techniques for the graph family
- More interesting: push techniques to their limits and beyond. “Embrace and Extend”

Stretching the definition of stretch

- Unweighted graphs: multiplicative and additive stretch
- Average stretch over edges, over pairs, other norms
- Average stretch of an edge over a distribution, other norms
- Partial, scaling, local, priority

Related problems

- In routing: labeled, name independent
- Distance oracles, distance labels
- Metric embedding
- Approximating a graph by a distribution of ultra metrics/spanning trees

Forbidden set routing

- Courcell's theorem: queries on bounded tree-width graphs can be answered in linear time
- Forbidden set routing for graphs of bounded tree-width [Twig 06]
  - Given source, destination labels, and a *set F of forbidden labels*, route along shortest path that avoids the set *F*
  - Stretch: $|route(x,y)| <= \textbf{stretch}\ dist_{G-F}(x,y)$
  - Stretch *1*, *poly-log* size labels and *poly-log* routing tables
- Forbidden label of *u*: distances in *G-u* between hubs (subgraph)
- Label s of *s,t*: distances in *G* between *s,t* and hubs (subgraph)
- Given a set *F* of forbidden labels and labels for *s,t*: build a *"sketch graph"* that combines all the safe edges of all the subgraphs. Then run shortest path on resulting sketch graph

Forbidden set routing

- Graphs with *bounded doubling dimension*
  - Stretch 1+eps, poly-log size labels, routing tables
- *Planar graphs*
  - Stretch 1+eps, poly-log size labels, routing tables
  - Hard problem – needed new shortest path separator details
  - Decompose into trees

From forbidden set routing to dynamic algorithms

- Adversary can add/remove nodes, ask for distance queries
- Build scheme in $O(n \text{ poly-log})$ time
- Accumulate $n^{1/2}$ changes, then rebuild scheme
- If processing each change costs poly-log then average cost is $O(n^{1/2} \text{ poly-log})$

- More details:
  - Adding a node: add to sketch graph the new node and its distances to all essential hubs
  - Remove a node: add forbidden label to sketch graph
- Do this $O(n^{1/2})$ times, then rebuild the whole scheme
- Query is $O(n^{1/2})$, can you decrease query cost?
- Open: better worst case bounds, for general graphs?
- What about dynamic compact routing?

Dynamic graph algorithms: current state

- Focus on general graphs, good amortized bounds, but hard to get good worst-case bounds
- Open: Single source, only remove nodes
- Perhaps general adversarial removals is too harsh. How do real world removals look like?
- Explore lower bounds
- Focus on when poly-log worst case is doable

Route planning

- Route planning: want stretch 1, fast queries and low memory
- 2010: several practical algorithms for route planning, no formal explanation why they work
- Highway dimension: a family of graphs in which these algorithms work well
  - Intuition and insight came from routing schemes with low doubling dimension
  - New ideas recently published… decomposing the problem into trees

- Hub labels: practical algorithm for computing exact distance labels. 10M nodes, 40M edges and labels of size 80 nodes
  - Intuition and insight came for routing and labeling schemes

- Dynamic algorithms with poly-log worst case bounds:
  - explains why practical algorithms work well, and how practical algorithms can guide theory

From compact routing to compact cutting

- Distance sparsifier (*Spanner*): a sparse subgraph that preserves a graph's distances
- The difference between distance oracles and distance labels/routing schemes
  - Total storage vs. per node storage
- Cut sparsifier : a sparse subgraph that preserves a graph's cut structure
  - Is there an analogue "cut labeling" scheme?
  - Total storage vs. per node storage
- Routing that looks at congestion

From compact routing to compact cutting

- Yiannis's sparcifier: decompose into trees
- Sample each edge with probability ½, double weight of surviving edges. Repeat *O(log n)* times
- Expected size of cut remains correct
  - Cuts with more than log n edges are fine due to concentration
- What about cuts with log n edges or less?
  - Decompose into trees
  - Peel graph with poly-log spanners

1. Compact routing: a fundamental distributed graph algorithm problem
2. General techniques: decompose into trees, sampling, partitioning
3. Forbidden-set routing: from monadic second order logic to compact routing to dynamic graph algorithms
4. Algorithm engineering: from compact routing on doubling graphs to hub labels and Bing maps
5. New directions: from compact routing to compact cutting. New connections between cuts, distances and spectral graph theory

“The important ideas of combinatorics do not usually appear in the form of precisely stated theorems, but more often as general principles of wide applicability”

The Two Cultures of Mathematics. Gowers

Three Messages

- (Try to) solve hard problems
- (Seek to) understand general principals
- Decompose graph problems into trees

Thank you