A Near-Optimal Distributed
Fully Dynamic Algorithm for
Maintaining Sparse Spanners

Michael Elkin

Ben-Gurion University
The Message-Passing Model

- $n$ processors reside in vertices of an unweighted undirected graph $G = (V, E)$. Each processor $v$ has a unique Id $I(v)$.

- Interconnected via links of $E$.

- Short messages ($O(\log n)$ bits).

- Unlimited computational power. Local computation requires zero time.
The Message-Passing Model (Cont.)

Synchronous setting (for this talk).

- Communication in *discrete* rounds.

- Messages sent in the beginning of a round $R$, arrive before the round $R + 1$ starts.

Running Time = $\#$ rounds.

Message Complexity = $\#$ messages.
Dynamic Model

Edges and vertices may appear or crash at will.

The weakest studied model.
(Weaker than controlled and partially controlled dynamic models.)

- Endpoints of a crashing edge are notified by a link-level protocol.

- A message is lost only if its edge crashes.

Motivation for the dynamic model: real-life networks, modern ad-hoc, sensor, wireless networks.

Primitive devices require simple algorithms!
Quiescence Complexity

Topology updates cease occurring at time $\alpha$. $\beta$ is the time when all vertices stop processing updates. At this point the algorithm maintains a correct structure.

Quiescence time $= \max\{\beta - \alpha\}$.
Quiescence message $= \#$ messages sent within $[\alpha, \beta]$. 
Spanners

Spanners = skeletons that approximate metric properties.

For $t \geq 1$, $G' = (V, H)$ is a \textit{t-spanner} of $G = (V, E)$, $H \subseteq E$, if $\forall u, w \in V$,

$$\text{dist}_{G'}(u, w) \leq t \cdot \text{dist}_G(u, w).$$
The Basic Tradeoff

[Peleg, Schaffer, 89]
∀ graph ∀t ∃ O(t)-spanner
with O(n^{1+1/t}) edges.

The best-known result
[Althofer, Das, Dobkin, Joseph, Soares, 90] -
(2t − 1)-spanner of size O(n^{1+1/t}).

An inherent tradeoff between the stretch parameter and the number of edges.

Optimal under Erdos girth conjecture.
Applications of Spanners

An underlying construct for many distributed algorithms.

- **Synchronization.**
  [Peleg,Ullman,89],
  [Awerbuch,Peleg,90]

- **Routing.**
  [Hassin,Peleg,99]

- **Approximate Distances and Shortest Paths Computation.**
  [Awerbuch,Berger,Cowen,Peleg,93],
  [Elkin,01]

- **Broadcast.**
  [Awerbuch,Goldreich,Peleg,Vainish,89],
  [Awerbuch,Baratz,Peleg,92]
Distributed Spanners

State-of-the-art distributed static algorithm.

[Baswana, Sen, 03],
[Baswana, Kavitha, Mehlhorn, Pettie, 05]

For $t = 1, 2, \ldots$, and $n$-vertex $G$, constructs $(2t - 1)$-spanner with expected $O(t \cdot n^{1+1/t})$ edges.

Time: $O(t)$.
Message: $O(|E| \cdot t)$.
Space: $O(deg(v) \cdot \log n)$.

Near-optimal tradeoff.
Dynamic State-of-the-Art

[Baswana, Sen, 03] composed with the simulation technique of [Awerbuch, Patt-Shamir, Peleg, Saks, 92]:

$(2t - 1)$-spanner of expected size $O(t \cdot n^{1+1/t})$,

Quiescence time: $O(t \cdot \log^3 n)$.

Quiescence message: $O(t \cdot |E| \cdot \log^3 n)$.

Space: $O(deg(v) \cdot \log^4 n)$.

Drawbacks of APSPS simulation technique:

Extremely complex (a reset procedure, neighborhood covers, a bootstrap technique, a local rollback).

Heavy local computations - unsuitable for simple devices.
Our Result

$(2t - 1)$-spanner of expected size $O(t \cdot n^{1+1/t})$.

Quiescence time: $3t$ instead of $O(t \cdot \log^3 n)$.
Note: $t \leq \log n$.

Quiescence message: worst-case $O(|E| \cdot t)$,
expected $O(|E|)$.
Space: $O(deg(v) \cdot \log n)$.
Expected local processing per edge: $O(1)$.

Lower bound: $2t/3$.
$t - 1$ under Erdos girth conjecture.

Better performance in purely incremental and purely decremental settings.

In both algorithms: non-adaptive adversary, oblivious to coin tosses.
Additional Features of our Algorithm

- **Treatment time:** If edges stop crashing at time $\alpha$, but are still allowed to appear, then at time $\alpha + 3t$ the spanner takes care of all edges present at time $\alpha$.

  Stronger than a bound on quiescence time!

- **Incremental setting:** bound of $2t$.

  If update set $F$ is a matching, quiescence time is 1!

- **Decremental setting:**

  If update set is of size $o(n^{1/t})$, the expected quiescence time is $1 + o(1)$. 
Historyless Dynamic Algorithm

**Standard approach**: maintain *history* of communication, undo operations based on the history.

Very expensive in terms of *local computation*. *Unfeasible* in wireless, sensor, ad-hoc networks.

**Our approach**: No history is stored!

Look for a “replacement” for crashing edges.

Undo operations, but the list-to-undo is deduced from the current state of affairs.

Reminiscent of *memoryless online* algorithms.
The Incremental Variant: Initialization

Focus on incremental algorithm.

Set a parameter $p \approx n^{-1/t}$.

Each $v$ picks a radius $r = r(v)$ from the truncated geometric distribution

$$\Pr(r = k) = p^k \cdot (1 - p), \text{ for } k \in [0, \ldots, t - 2],$$
and
$$\Pr(r = t - 1) = p^{t-1}.$$

Memoryless distribution

$$\Pr(r \geq k + 1 \mid r \geq k) = p$$
for $k \in [0, 1, \ldots, t - 2]$.

[Linial, Saks, 92], [Bartal, 96]
Labels

Each $v$ has a unique id $I(v)$, and a label $P(v) = (B(P(v)), L(P(v)))$.

Initially, $P(v) \leftarrow (I(v), 0)$.

$P = (B(P), L(P))$.

Implicitly, the algorithm maintains a tree cover. (A set of not necessarily disjoint trees that cover all vs.)

$B(P)$ - the id of a tree $\tau$ to which the vertex $v$ labeled by $P$ currently belongs.

$L(P)$ - the distance between $v$ and the root of $\tau$.

The vertex $w = w_P$ s.t. $I(w) = B(P)$ is the base vertex of $P$.

$w_P$ is the root of the tree $B(P)$. 

+ 15
\( r(w_P) \) - maximum distance to which \( B(P) = I(w_P) \) is allowed to propagate. The tree \( B(P) \) cannot be deeper than \( r(w_P) \).

\[ \Rightarrow \quad \text{For each label } P, \quad L(P) \leq r(w_P). \]

A label \( P \) is selected if \( L(P) < r(w_P) \). In this case \( v \) may be an internal vertex of the tree \( B(P) \).

For a label \( P \),
\[ \text{IP}(P \text{ is selected}) = \]
\[ = P(r(w_P) \geq L(P) + 1 \mid r(w_P) \geq L(P)) \leq \]
\[ \leq p \approx 1/n^{1/t}. \]

Probability of a label to reach level \( t - 1 \) is
\[ \text{IP}(r = t - 1) = \text{IP}(r = t - 1 \mid r \geq t - 2). \]
\[ \text{IP}(r \geq t - 2 \mid r \geq t - 3) \cdot \ldots \cdot \text{IP}(r \geq 1 \mid r \geq 0) = \]
\[ = p^{t-1} \approx \left(\frac{1}{n^{1/t}}\right)^{t-1}. \]

Hence, whp, the \#labels of level \( t - 1 \)
\[ \approx n^{1/t}. \]
Comparing labels:

\[ P(v) \succ P(v') \text{ iff either} \]
\[ (L(v), B(v)) > (L(v'), B(v')) \text{ or} \]
\[ (((L(v), B(v)) = (L(v'), B(v'))) \land (I(v) > I(v'))) \].

Vertices adopt labels from their neighbors. When \( v \) adopts a label from \( u \), it becomes its child in the tree \( B(P) \), \( P = P(u) \). When a label \( P \) is adopted, \( L(P) \) is incremented, but \( B(P) \) stays unchanged.
Data Structures

Every $v$ maintains an edge set $Sp(v)$.

Initially, $Sp(v) = \emptyset$.

$Sp(v)$ grows monotonely.

$Sp(v) = T(v) \cup X(v)$.

$T(v)$ - the tree edges of $v$.

$X(v)$ - the cross edges of $v$.

An implicit construction of a tree cover. Edges of the tree cover are stored in $T(v)$’s.

The spanner also has edges connecting different trees. Those are edges of $X(v)$’s.
For each vertex $v$, the algorithm maintains a table $M(v)$. Initially, $M(v) = \emptyset$.

$M(v)$ is the set of trees to which $v$ is already connected in the spanner.

$e' = (v,z) \text{ in } X(v) \implies B(P(z)) \text{ in } M(v)$

$B(P(z)) = B(P(u))$

\[\downarrow\downarrow\]

$e \text{ can be dropped!}$
The Algorithm
(for a vx v)

For 2t rounds from the beginning or after detecting a new edge do

Go over all received messages and do
while ∃ message $P(u)$ with $P(u) \succ P(v)$

// adopt the label of $u$
if $P(u)$ is selected
    $B(P(v)) \leftarrow B(P(u));$
    $L(P(v)) \leftarrow L(P(u)) + 1;$
    add $(v,u)$ to $S_p(v);$ // to $T(v)$
else if $B(P(u)) \not\in M(v)$
    add $B(P(u))$ to $M(v);$
    add $(v,u)$ to $S_p(v);$ // to $X(v)$
end-if
end-while
Send to all neighbors the message $P(v)$.

Remark: Testing whether $P(u)$ is selected is done by standard techniques.
The Algorithm: Discussion

Very simple:

1. One type of messages.

2. The same behavior on each round.

3. A handful of local variables.

4. Basic data structures.
Definition - Scanning an Edge

\( v \text{ scans } e = (v, u) \) if \( P(u) \) passes
the while-loop condition of the vertex \( v \).

It may happen that on a given round, neither \( v \) nor \( u \) scan the edge \((v, u)\).
(Due to different order in which \( v \) and \( u \) process edges.)
Example

At the beginning of a round, 
\( P(v) \succ P(u). \)

The vertex \( v \) considers the message \( P(u) \) before all other messages, and discovers that \( P(v) \succ P(u). \)
Thus \( v \) does not scan \( e \).

The vertex \( u \) considers first another message \( P(z). \)
As a result \( u \) increases its label to \( P'(u) \succ P(v) \succ P(u), \) and then considers the message \( P(v). \)

However, since \( P'(u) \succ P(v), \) it does not scan \( e \) either.

So, neither \( v \) nor \( u \) scan \( e \) on this round!
Analysis - Scanning Edges

**Lm:** Every edge \( e = (v, u) \) is eventually scanned.

**Pf:** \( v \) increases its label \( \leq t - 1 \) times.

The same applies for \( u \).

Hence among \( 2t \) rounds

\( \exists \) round (other than the first)
on which neither \( v \) nor \( u \) increase their labels.

On this round either \( v \) or \( u \) scan \( e \).

QED
**Analysis - Stretch**

**Lm:** Suppose $v$ was labeled by $P$ at some point.
Then $\exists$ path between $w_P$ and $v$ of length $\leq L(P)$ in $\bigcup_{z \in V} T(z)$.

**Pf:** Induction on $L(P)$.

For $v$ to get label $P$, it must have inserted an edge $(v, u)$ into $T(v)$, s.t. $L(P(u)) = L(P) - 1$, $B(P(u)) = B(P)$.

The induction hypothesis is applicable to $u$.
QED

**Cor:** If $v$ used to be labeled by $P$, and $v'$ by $P'$, and $B(P) = B(P')$, then $\exists$ path of length $\leq L(P) + L(P') \leq 2t - 2$ between $v$ and $v'$ in $\bigcup_{z \in V} T(z)$. 

+ 23
Lm: If $v$ scans $e = (v, u)$, from that point on $\exists$ path of length $\leq 2t - 1$ between $v$ and $u$ in the spanner.

Pf:
If $((P(u) \text{ is selected})$ or $(B(P(u)) \notin M(v)))$, $e \in Sp(v)$, and we are done.

If $((P(u) \text{ is not selected})$ and $(B(P(u)) \in M(v)))$, $\exists u'$ s.t. $u'$ used to have label $P'$ with $B(P') = B(P(u))$, and $e' = (v, u') \in X(v) \subseteq Sp(v)$.

$\Downarrow$

$\exists uu'$-path of length $\leq 2t - 2$ in the spanner, and
$\exists uv$-path of length $\leq 2t - 1$. QED
Local Processing

$\tilde{O}(t \cdot n^{1/t})$ space.

Maintain a data structure of $\tilde{O}(t \cdot n^{1/t})$ base values.

Support existence and insertion queries.

Naively:
\[
\log \tilde{O}(t \cdot n^{1/t}) = O\left(\frac{\log n}{t} + \log \log n\right)
\]
time-per-query whp.
(a balanced search tree (BST))

More sophisticatedly:
\[o(\log \log n)\] time-per-query whp.
(hash + BST in each entry).

Open question: $O(1)$ whp?
Summary

- Optimal solution for the dynamic distributed spanner problem.

- Historyless paradigm for devising dynamic distributed algorithms.

- Lower bound of $\Omega(t)$.

- Streaming algorithm.

- Centralized dynamic algorithm.
Open Questions

• Applications for the dynamic distributed spanners: Synchronization (?), Routing (?), Online load balancing (?).

• Applications for the historyless paradigm.

• Achieve $O(1)$ processing time-per-edge whp.

• Achieve spanner size of $O(n^{1+1/t})$ instead of $O(t \cdot n^{1+1/t})$ (even for static model).

• Derandomize.
  Less challenging - devise algorithm for an adaptive adversary, or for a non-oblivious one.