# Dynamic Shortest Paths

Giuseppe F. Italiano
*University of Rome "Tor Vergata"*

PATH05 Summer School on Shortest Paths
*Copenhagen, July 4-8, 2005*

## Outline

- Dynamic Graph Problems
- State of the Art
- Algorithmic Techniques
- Conclusions

## Static Graphs…

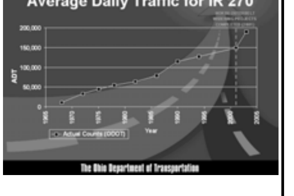

Paulus Ritius - *Portae lucis*, Augsburg, 1516.

Graphs have been used for centuries to model relationships in life…

A static life?

## …or Dynamic Graphs?

Sometimes, life looks a bit more dynamic …




## Dynamic Graphs

Graphs subject to **update** operations

Typical updates:
```
Insert(u,v)
Delete(u,v)
SetWeight(u,v,w)
```

## Dynamic Graphs

Partially dynamic problems

Graphs subject to insertions only, or deletions only, but not both.

Fully dynamic problems

Graphs subject to intermixed sequences of insertions and deletions.

## Dynamic Graph Problems

Support **query** operations about certain property on a dynamic graph

- *Dynamic Connectivity (undirected graph G)*
  `Connected(x,y)`:
  are x and y connected in G?
- *Dynamic Transitive Closure (directed graph G)*
  `Reachable(x,y)`:
  is y reachable from x in G?

## Dynamic Graph Problems

- *Dynamic All Pairs Shortest Paths*
  `Distance(x,y)`:
  what is the distance from x to y in G?
  `ShortestPath(x,y)`:
  what is the shortest path from x to y in G?
- *Dynamic Minimum Spanning Tree (undirected graph G)*

  any property on a MST of G

## Dynamic Graph Problems

- *Dynamic Min Cut*
  `Cut(x)`:
  what side of a global minimum cut of G x belongs to?
- *Dynamic Planarity Testing*
  `planar()`:
  is G planar?
- *Dynamic k-connectivity*
  `k-connected()`:
  is G k-connected?

## Dynamic Graph Algorithms

The goal of a dynamic graph algorithm is to support **query** and **update** operations as quickly as possible.

*We will sometimes use amortized bounds:*

$$\frac{\text{Total worst-case time over sequence of ops}}{\text{\# operations}}$$

*Notation:*
G = (V,E)
n = |V|
m = |E|

## Fully Dynamic APSP

Given a weighted directed graph G = (V,E,w), perform any intermixed sequence of the following operations:

| | |
|---|---|
| Update(v,w): | update edges incident to v [w( )] |
| Query(x,y): | return distance from x to y (or shortest path from x to y) |

## Outline

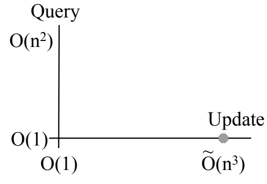
- Dynamic Graph Problems
- **State of the Art** 
- Algorithmic Techniques
- Conclusions

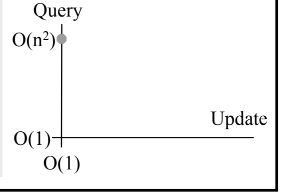
## Simple-minded Approaches

| Fast query approach | Fast update approach |
|---|---|
| Keep the solution up to date. Rebuild it from scratch at each update. | Do nothing on graph. Visit graph to answer queries. |

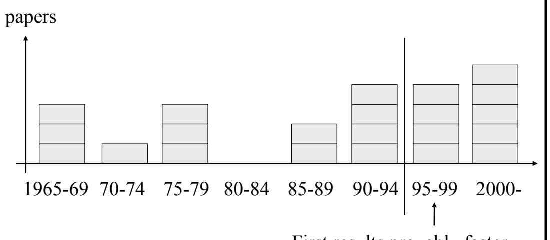
## Dynamic All-Pairs Shortest Paths

| Fast query approach | Fast update approach |
|---|---|
| Rebuild the distance matrix from scratch after each update. | To answer a query about (x,y), perform a single-source computation from x. |




## Previous Work on Dynamic APSP




First results provably faster than recomputing from scratch in the late 90s

## State of the Art

First fully dynamic algorithms date back to the 60's

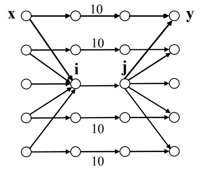
Until 1999, none of them was better in the worst case than recomputing APSP from scratch (~ cubic time!)

- P. Loubal, A network evaluation procedure, ***Highway Research Record 205***, 96-109, **1967**.
- J. Murchland, The effect of increasing or decreasing the length of a single arc on all shortest distances in a graph, TR LBS-TNT-26, Transport Network Theory Unit, London Business School, **1967**.
- V. Rodionov, A dynamization of the all-pairs least cost problem, ***USSR Comput. Math. And Math. Phys. 8***, 233-277, **1968**.
- …

| | Graph | Weight | Update | Query |
|---|---|---|---|---|
| Ramalin.&Reps 96 | general | real | $O(n^3)$ | O(1) |
| King 99 | general | [0,C] | $O(n^{2.5} (C \log n)^{0.5})$ | O(1) |

## Fully Dynamic APSP

Edge insertions (edge cost decreases)




For each pair x,y check whether

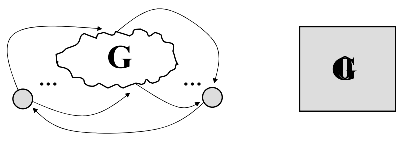
$D(x,i) + w(i,j) + D(j,y) < D(x,y)$

Quite easy: $O(n^2)$

## Fully Dynamic APSP

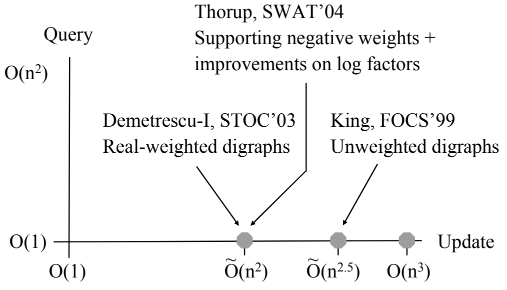
- Edge deletions (edge cost increases)

  Seem the hard operations. Intuition:




- When edge (shortest path) deleted: need info about second shortest path? (3rd, 4th, …)

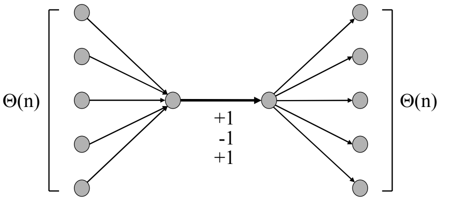
## Recent progress in dynamic APSP




Decremental bounds: Baswana, Hariharan, Sen STOC'02

Approximate dynamic APSP: Roditty, Zwick FOCS'04

## Quadratic Update Time Barrier?




If distances are to be maintained explicitly, any algorithm must pay $\Omega(n^2)$ per update…

## Related Problems

*Dynamic Transitive Closure (directed graph G)*

| update | query | authors | notes |
|---|---|---|---|
| $O(n^2 \log n)$ | $O(1)$ | King, FOCS'99 | |
| $O(n^2)$ | $O(1)$ | King-Sagert, STOC'99 | *DAGs* |
| | | Demetrescu-I., FOCS'00 | |
| | | Roditty, SODA'02 | |
| | | Sankowski, FOCS'04 | worst-case |
| $O(n^{1.575})$ | $O(n^{0.575})$ | Demetrescu-I., FOCS'00 | *DAGs* |
| | | Sankowski, FOCS'04 | |
| $O(m\, n^{1/2})$ | $O(n^{1/2})$ | Roditty, Zwick, FOCS'02 | |
| $O(m+n \log n)$ | $O(n)$ | Roditty, Zwick, FOCS'04 | |

Decremental bounds: Baswana, Hariharan, Sen, STOC'02

## Other Problems

*Dynamic Connectivity (undirected graph G)*

| update | query | authors |
|---|---|---|
| $O(\log^3 n)$ | $O(\log n/\log \log n)$ | Henzinger, King, STOC'95 (randomized) |
| $O(n^{1/3} \log n)$ | $O(1)$ | Henzinger, King, ICALP'97 |
| $O(\log^2 n)$ | $O(\log n/\log \log n)$ | Holm, de Lichtenberg, Thorup, STOC'98 |

Lower bounds:
$\Omega(\log n)$ update bound by Patrascu and Demaine, STOC'04

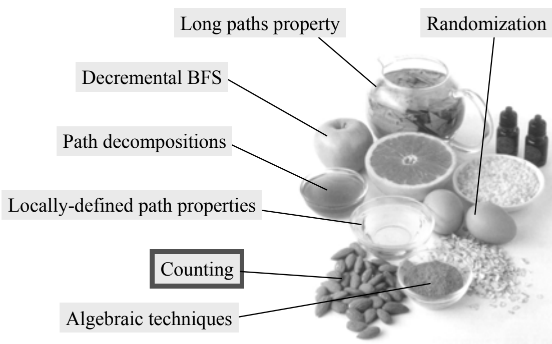
## Outline

- Dynamic Graph Problems
- State of the Art
- **Algorithmic Techniques** 
- Conclusions

## Algorithmic Techniques

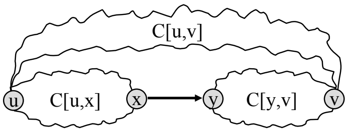
Will focus on techniques for path problems.
Running examples: shortest paths/transitive closure

## Main Ingredients




## Path Counting [King/Sagert, STOC'99]

Maintain reachability information by keeping a count of the number of distinct paths in acyclic digraphs




$\forall$ u,v: $C[u,v] \leftarrow C[u,v] + C[u,x] \cdot C[y,v]$ $O(n^2)$

Problem: counters as large as $2^n$

Solution: use arithmetic modulo a random prime…

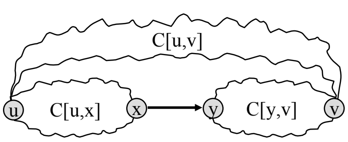
## Dynamic Transitive Closure [King/Sagert, STOC99]

Update: $O(n^2)$ w.c. time
Query: $O(1)$ w.c. time

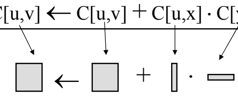
*For acyclic digraphs.*
*Randomized, one-sided error.*

Can we trade off query time for update time?

## Looking from the matrix viewpoint



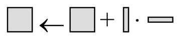

$\forall$ u,v: $C[u,v] \leftarrow C[u,v] + C[u,x] \cdot C[y,v]$



## Maintaining dynamic integer matrices

Given a matrix M of integers, perform any intermixed sequence of the following operations:

`Update`(J,I): $M \leftarrow M + J \cdot I$ $O(n^2)$



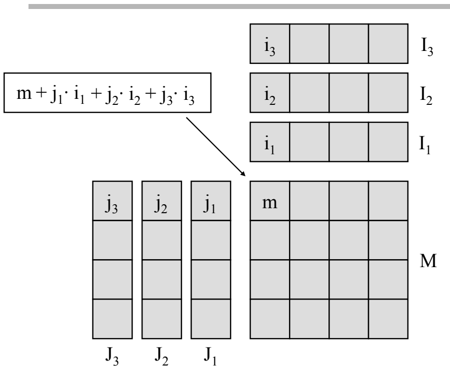
`Query`(i,j): return M[i,j] $O(1)$

## Maintaining dynamic integer matrices

How can we trade off operations?

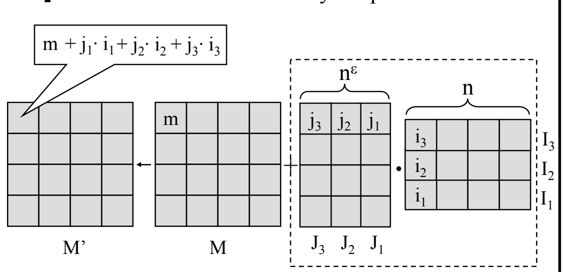
Lazy approach: buffer at most $n^\varepsilon$ updates
Global reconstruction every $n^\varepsilon$ updates
Reconstruction done via matrix multiplication

## Maintaining dynamic integer matrices



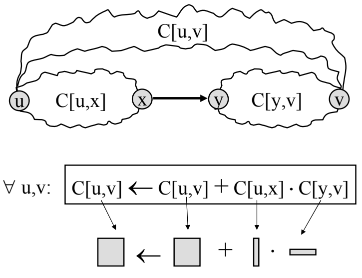

## Maintaining dynamic integer matrices

Global reconstruction every $n^\varepsilon$ updates

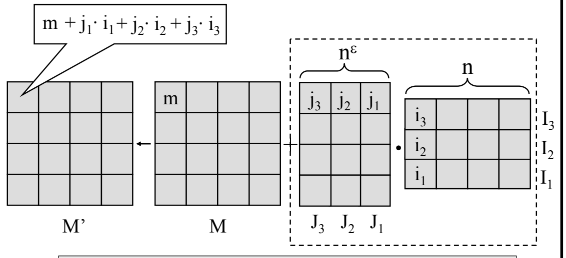



$O(n^{\omega(1,\varepsilon,1)})$

## Back to Dynamic Transitive Closure



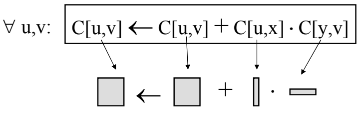

## Query Time




Total Query time $O(n^{\varepsilon})$

## Update Time

1. Compute C[u,x] and C[y,v] for any u,v

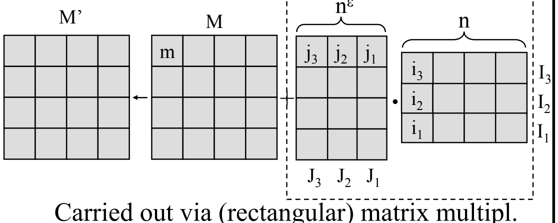



Carried out via O(n) queries

Time: $O(n^{1+\varepsilon})$

## Update Time

2. Global rebuild every $n^{\varepsilon}$ updates




Carried out via (rectangular) matrix multipl.

Amortized time: $O( n^{\omega(1,\varepsilon,1)} / n^{\varepsilon})$

## Dynamic Transitive Closure [Demetrescu-I., FOCS'00]

Update: $O(n^{\omega(1,\varepsilon,1)-\varepsilon}+n^{1+\varepsilon})$
Query: $O(n^{\varepsilon})$
for any $0 < \varepsilon < 1$

Find $\varepsilon$ such that $\omega(1,\varepsilon,1) = 1+2\varepsilon$

Best bound for rectangular matrix multiplication [Huang/Pan98]

⇩

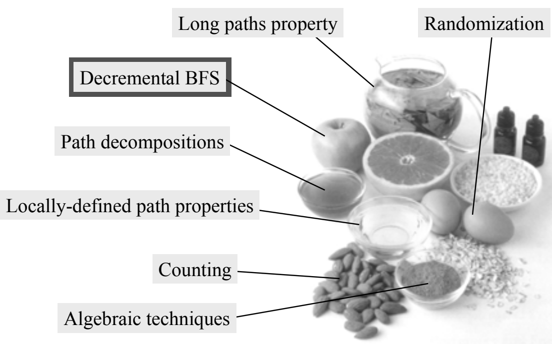
$\varepsilon < 0.575$

Update: $O(n^{1.575})$ worst-case time
Query: $O(n^{0.575})$ worst-case time

## Exercise 1

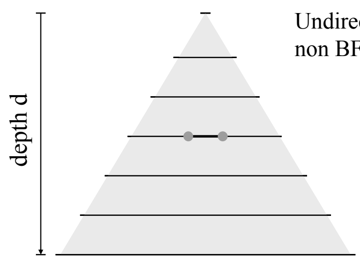
Why is this working for acyclic graphs only?

## Main Ingredients




## Decremental BFS [Even-Shiloach, JACM'80]

Maintain BFS levels under deletion of edges

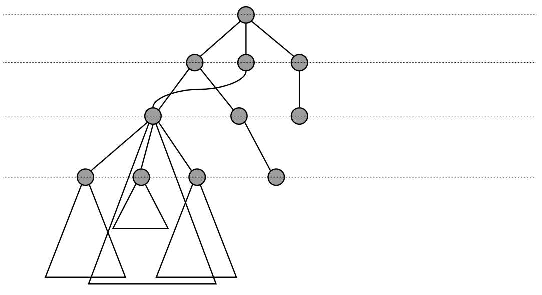



Undirected graphs: non BFS-tree edges can be

either between two consecutive levels

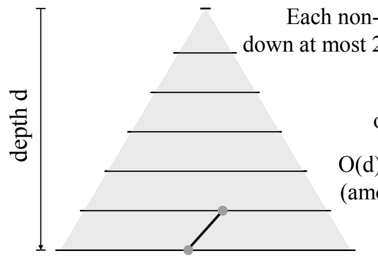
or at the same level

## Decremental BFS [Even-Shiloach, JACM'80]



## Decremental BFS [Even-Shiloach, JACM'80]

This implies that during deletion of edges




Each non-tree edge can fall down at most 2d times overall…

O(md) total time over any sequence

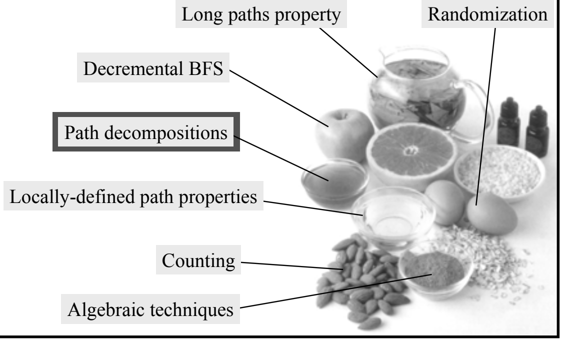
O(d) time per deletion (amortized over Ω(m) deletions)

## Exercise 2 and 3

Exercise 2: Extend this to unweighted directed graphs.

Exercise 3: Extend this to directed graphs with integer weights.

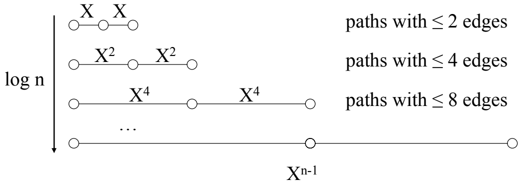
## Main Ingredients



## Doubling Decomposition [folklore]

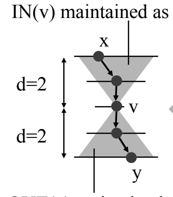
Transitive closure can be computed with O(log n) products of Boolean matrices

X = adjacency matrix + I    $X^{n-1}$ = transitive closure



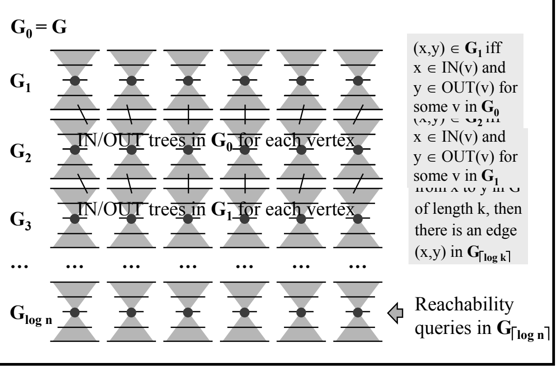

## Dynamic Transitive Closure [King, FOCS'99]

Ingredients: Decremental BFS + Doubling decomposition



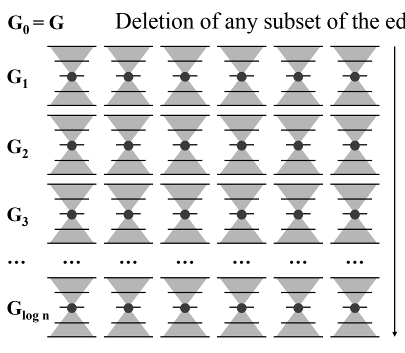

Total cost for building the two trees + deleting all edges: O(m)

## Dynamic Transitive Closure [King, FOCS'99]




## Dynamic Transitive Closure [King, FOCS'99]

Deletion of any subset of the edges of G

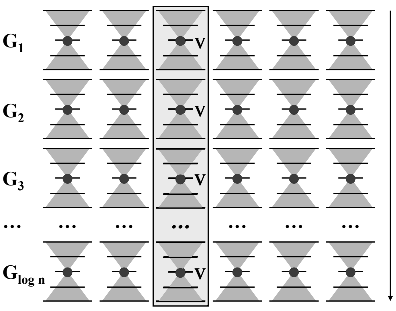



Edge deletions

(cost charged to the creation of the trees)

## Dynamic Transitive Closure [King, FOCS'99]

$G_0 = G$ Insertion of edges incident to a vertex v

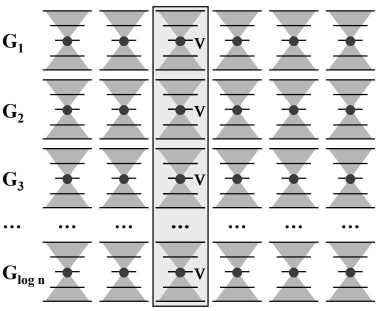



IN(v) and OUT(v) rebuilt from scratch on each level…

O($n^2$ log n) time

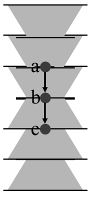
## Dynamic Transitive Closure [King, FOCS'99]

$G_0 = G$ Insertion of edges incident to a vertex v

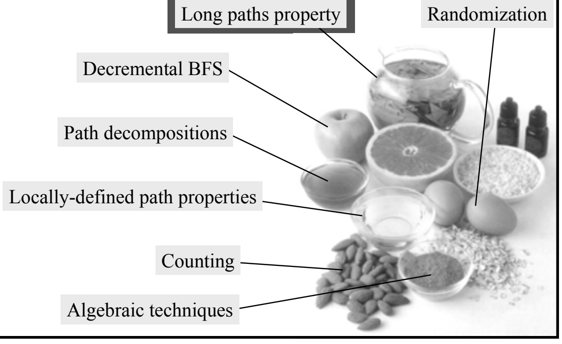



Correctness?

Path a,b,c in $\mathbf{G_{i-1}}$ $\Rightarrow$ (a,c) in $\mathbf{G_i}$ ?



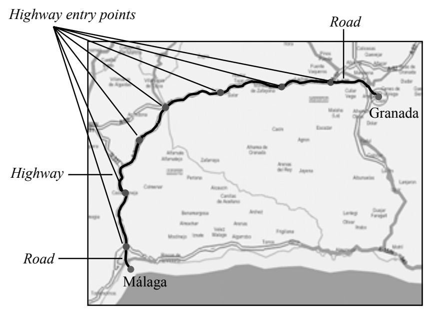

## Main Ingredients




## A real-life problem

## Why Granada?

## Decomposition of long paths




Are there roads and highways in graphs?

## Long Paths Property [Greene-Knuth '82]

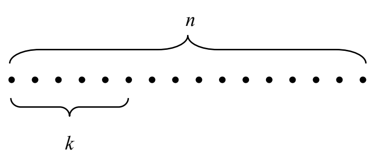
Let $\boldsymbol{P}$ be a path of length at least $\boldsymbol{k}$.

Let $\boldsymbol{S}$ be a random subset of vertices of size $(c\, n \ln n) / k$.

Then with high probability $P \cap S \neq \varnothing$.

Probability $p \geq 1 - 1/n^c$

## Long Paths Property [Greene-Knuth '82]




Select each element independently with probability $p = \frac{c \ln n}{k}$

The probability that a given set of $\boldsymbol{k}$ elements is not hit is $(1-p)^k = \left(1 - \frac{c \ln n}{k}\right)^k < n^{-c}$

## Long Paths Property [Greene-Knuth '82]

Let $P$ be a path of length at least $k$.

Let $S$ be a random subset of vertices of size $(c\, n \ln n) / k$.

Then with high probability there is no subpath of $P$ of length $k$ with no vertices in $S$ ($P \cap S \neq \varnothing$).

Probability $p \geq 1 - 1 / n^{\alpha(c)}$ for some $\alpha > 0$.

## Exercise 4

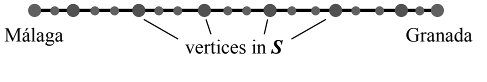
Prove the stronger form of the long paths property.

## Long Paths Property

Randomly pick a set $S$ of vertices in the graph

$$|S| = \frac{c\, n \log n}{k} \qquad c, k > 0$$

Then on any path in the graph every $k$ vertices there is a vertex in $S$, with probability $p \geq 1 - 1 / n^{\alpha(c)}$

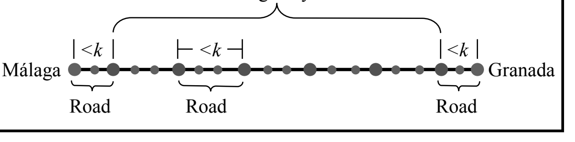



## Roads and Highways in Graphs

Highway entry points = vertices in $S$

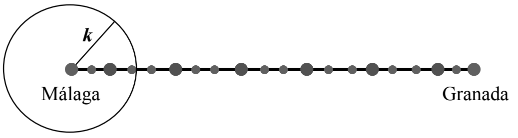
Road = shortest path using at most $k$ edges

Highway = shortest path between two vertices in $S$



## Computing Shortest Paths 1/3

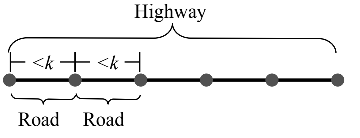
1 Compute roads
(shortest paths using at most ***k*** edges)




Even & Shiloach BFS trees may become handy…

## Computing Shortest Paths 2/3

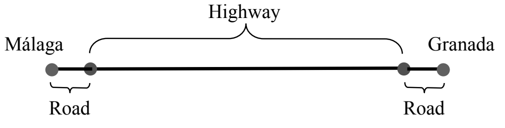
2 Compute highways
(by stitching together roads)




…essentially an all pairs shortest paths computation on a contracted graph with vertex set ***S***, and edge set = roads

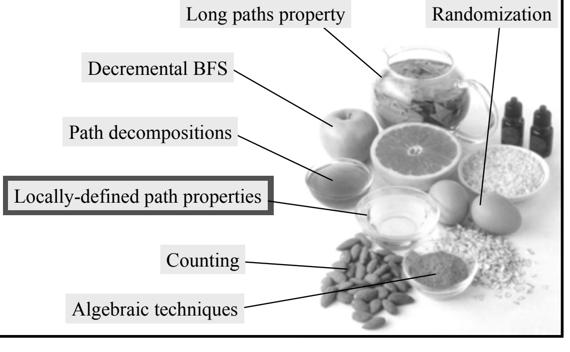
## Computing Shortest Paths 3/3

3 Compute shortest paths (longer than ***k*** edges)
(by stitching together roads + highways + roads)



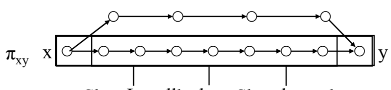

Used (for dynamic graphs) by King [FOCS'99], Demetrescu-I. [ICALP'02], Roditty-Zwick [FOCS'04], …

## Main Ingredients



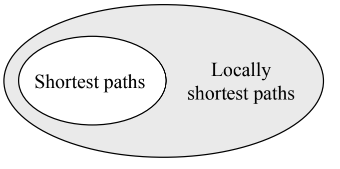
## Locally Shortest Paths [Demetrescu-I., STOC'03]

A path is *locally shortest* if all of its **proper** subpaths are shortest paths



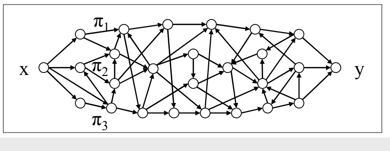

## Locally shortest paths [D-Italiano, STOC'03]

By optimal-substructure property of shortest paths:




## Properties of locally shortest paths

*Property 1*
Locally shortest paths $\pi_{xy}$ are internally vertex-disjoint

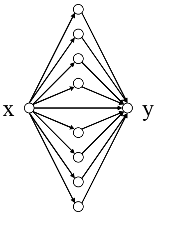



This holds under the assumption that there is a unique shortest path between each pair of vertices in the graph

(Ties can be broken by adding a small perturbation to the weight of each edge)

## Properties of locally shortest paths

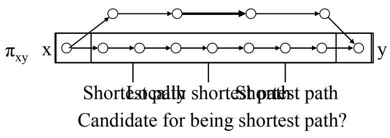
*Property 2*
There can be at most n-1 LS paths connecting x,y




This is a consequence of vertex-disjointess…

## Locally shortest paths for dynamic APSP

- Hard operations seem edge deletions (edge cost increases)
- When edge (shortest path) deleted: need info about second shortest path? (3rd, 4th, …)
- Hey… what about locally shortest paths?




## Locally shortest paths for dynamic APSP

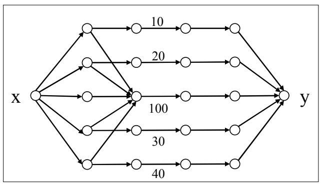
Idea: Maintain all the locally shortest paths of the graph

How do locally shortest paths change in a dynamic graph?

## Appearing locally shortest paths

***Fact 1***

At most mn ($n^3$) paths can **start** being locally shortest after an edge weight increase




## Disappearing locally shortest paths

***Fact 2***

At most $n^2$ paths can **stop** being locally shortest after an edge weight increase

$\pi$ stops being locally shortest after increase of ***e***

subpath of $\pi$ (was shortest path) must contain ***e***

shortest paths are unique: at most $n^2$ contain ***e***

## Maintaining locally shortest paths

# Locally shortest paths appearing after increase: $< n^3$

# Locally shortest paths disappearing after increase: $< n^2$

The amortized number of changes in the set of locally shortest paths at each update in an increase-only sequence is $O(n^2)$

## An increase-only update algorithm

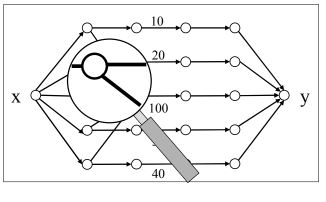
This gives (almost) immediately:

$O(n^2 \log n)$ amortized time per increase

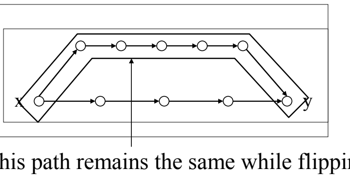
$O(mn)$ space

## Maintaining locally shortest paths

What about fully dynamic sequences?

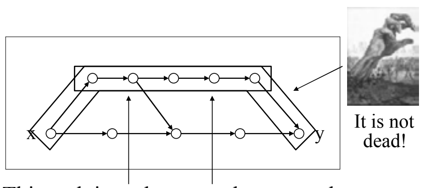



## How to pay only once?




This path remains the same while flipping between being LS and non-LS:
Would like to have update algorithm that pays only once for it until it is further updated...

## Looking at the substructure




This path is no longer a shortest path after the insertion…

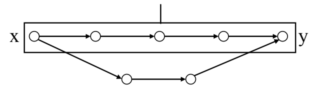
This path remains a shortest path after the insertion

…but if we removed the same edge it would be a shortest path again!

## Historical paths

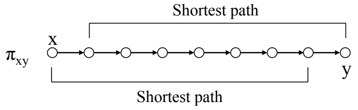
A path is **historical** if it was shortest at some time since it was last updated

historical path

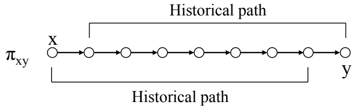



## Locally historical paths

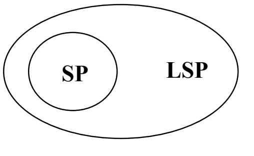
Locally shortest path



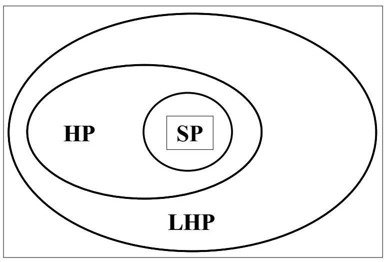

Locally historical path



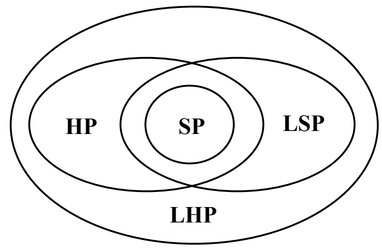

## Key idea for partially dynamic



## Key idea for fully dynamic




## Putting things into perspective…




## The fully dynamic update algorithm

Idea: Maintain all the locally historical paths of the graph

Fully dynamic update algorithm very similar to partially dynamic, but maintains locally historical paths instead of locally shortest paths (+ performs some other operations)

$O(n^2 \log^3 n)$ amortized time per update

$O(mn \log n)$ space

## Further Improvements [Thorup, SWAT'04]

Using locally historical paths, Thorup has shown:

$O(n^2 (\log n + \log^2 (m/n)))$ amortized time per update

$O(mn)$ space

## Exercise 5

"Il Gugol", a reowned Italian Web search engine, recently bought one 20 GB RAM machine to store its snapshot of the Web graph, having 1 million nodes and 10 million edges. Il Gugol runs state-of-the-art fully dynamic all-pairs shortest paths algorithms on this graph. Will the machine be powerful enough?

## Outline

- Dynamic Graph Problems
- State of the Art
- Algorithmic Techniques
- **Conclusions**

## More Work to do on Dynamic APSP

- Space is a BIG issue in practice
  (talk to people in "Il Gugol")
- More tradeoffs for dynamic shortest paths?
  Roditty-Zwick, ESA 04 $\tilde{O}(mn^{1/2})$ update, $O(n^{3/4})$ query for unweighted
- Worst-case bounds?
  Thorup, STOC 05 $\tilde{O}(n^{2.75})$ update
- Lower bounds?

## Some Open Problems…

- *Fully Dynamic Single-Source Shortest Path*
  Nothing better than simple-minded approaches
- *General Techniches for making increase-only fully dynamic?*
  Fully exploited on dynamic undirected graphs

## Some Open Problems…

- *Dynamic Maximum st-Flow*
  `Flow(x,y)`:
  what is the flow assigned to edge (x,y) in a maximum st-flow in G?
- *Dynamic Diameter*
  `Diameter()`:
  what is the diameter of G?
- *Dynamic Strongly Connected Components (directed graph G)*
  `SCC(x)`:
  what is the representative vertex in the SCC of G that contains x?