Lower Bounds for Non-Adaptive Shortest Path Relaxation

David Eppstein

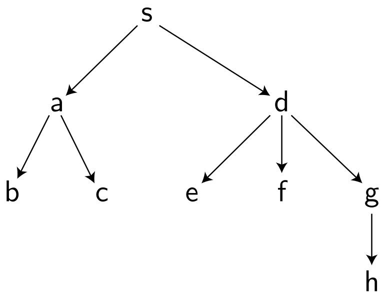
WADS 2023

Single source shortest path problem

Input: directed graph with edge lengths plus starting vertex s

Outputs

- Tree of shortest paths from s to all other reachable vertices
- Distances (lengths of paths) to all vertices ($+\infty$ if unreachable)



Represent output by two decorations for each vertex x:

$$P[x] = \text{parent vertex of } x$$
$$D[x] = \text{distance from start vertex to } x$$

Time bounds for single-source shortest paths

Assume: Input may have negative edges but no negative cycles
(otherwise shortest simple path is NP-hard)

With integer edge weights in range $[-W, W]$:
Randomized $O(m \log^2 n \log nW)$
[Bernstein et al. 2022; Bringmann et al. 2023]

Main idea: Recursive low-diameter decomposition

Best strongly polynomial time bound known:
$O(mn)$ for the Bellman–Ford algorithm

Main idea: Relaxation

Relaxation algorithms (more detail)

Initialize: $P[x] =$ None; $D[x] = 0$ if $x = s$, $+\infty$ otherwise

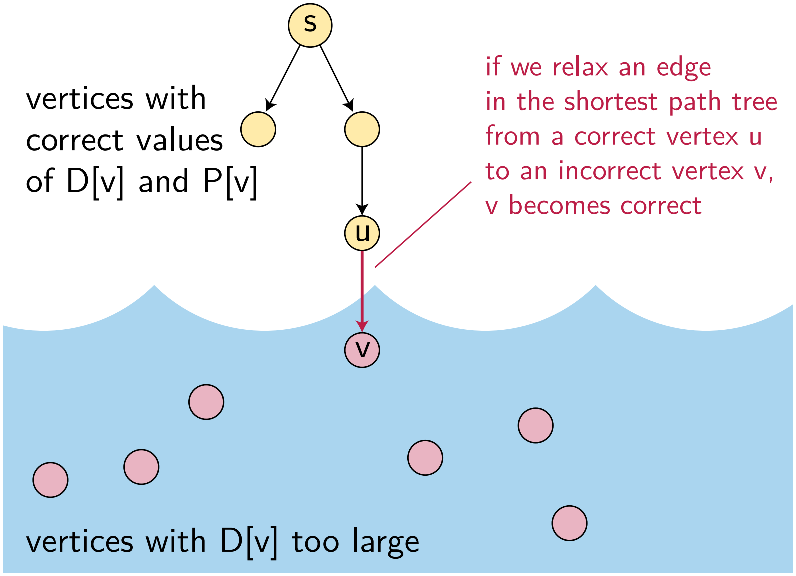
"Relax" edge uv: test whether path to u + edge uv gives a better path to v, and if so update the decorations for v

```
def relax(u,v):
    if D[u] + length(edge uv) < D[v]:
        D[v] = D[u] + length(edge uv)
        P[v] = u
```

Key insights:

- Initialization gives s the correct decorations (its distance and parent in the actual shortest path tree)
- If shortest path to v goes through edge uv and u already has correct decorations, then relax(uv) gives v correct decorations
- Other calls to relax are harmless (maintain invariant that $D[v] \geq$ actual distance)

Intuitive picture of a relaxation algorithm


s
u
v
vertices with correct values of D[v] and P[v]
if we relax an edge in the shortest path tree from a correct vertex u to an incorrect vertex v, v becomes correct
vertices with D[v] too large

Examples of relaxation algorithms

Directed acyclic graphs: relax in a topological ordering

m relaxations

Dijkstra: relax, ordered by tentative distance

m relaxations, $O(m + n \log n)$ overhead

Bellman–Ford (unoptimized): relax all edges repeatedly, $n - 1$ times

$m(n-1)$ relaxations

Bellman–Ford (optimized): partition into two acyclic subgraphs wrt random vertex order, relax each subgraph in topological order, repeat until no more changes

$\approx mn/3$ relaxations [Bannister and Eppstein 2012]

How low can we go?

For worst-case graphs (allowing cycles, and negative edge weights, but not negative cycles) how few relaxation steps are needed?

Not a well-posed question

Find shortest paths some other way, not involving relaxation, e.g. recent near-linear-time algorithms for graphs with small integer edge weights [Bernstein et al. 2022]

Once you know the shortest path tree, apply the DAG algorithm to that tree, ignoring the rest of the graph $\Rightarrow$ $n - 1$ relaxations

Focus on relaxation

To avoid cheating, study non-adaptive relaxation algorithms:

- Using graph structure but not edge weights, determine sequence of edges to relax
- Relax the edges in that order

Examples of non-adaptive algorithms:

- Directed acyclic graph algorithm
- Unoptimized Bellman–Ford
- Optimized randomized Bellman–Ford with $n/3 + o(n)$ repetitions (correct w.h.p.)
- Bellman–Ford-orderable graphs: relax each edge once in a fixed order w/ guaranteed correctness [Haddad and Schäffer 1988]

Main results

Number of relaxation steps of non-adaptive relaxation algorithms, for single-source shortest paths on directed graphs, must be $\geq$:

- $\left(\frac{1}{6} - o(1)\right)n^3$, for deterministic algorithms on complete graphs
- $\left(\frac{1}{12} - o(1)\right)n^3$, for randomized algorithms on complete graphs (required to be correct with high probability)
- $\Omega\left(\dfrac{mn}{\log n}\right)$ on sparse graphs, deterministic or randomized
- $\Omega(mn)$, when $m = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$

Shorter summary: Bellman–Ford is optimal or near-optimal

Main idea: Deterministic, complete graphs

A correct relaxation sequence must have a subsequence that relaxes the edges of each shortest path in order

Adversary (knowing relaxation sequence) chooses Hamiltonian path, greedily, 2 steps at a time, so 2nd edge is relaxed as late as possible

(Chosen edges get low weight; unchosen get high weight)

First step must connect to the part of the path that was already chosen but the second step can be any disjoint edge

# relaxations before sequence reaches chosen edge
$\geq$ # disjoint edges remaining to choose
$\approx$ square of # vertices remaining

Main idea: Random, complete graphs

Replace length of high-probability-correct relaxation sequence by expected length of path subsequence in too-long relaxation seq

(Doesn't change length much, easier to analyze)

Two-player game: relaxer chooses sequence, adversary chooses path, to min- or maximize subseq length

Yao minimax principle [Yao 1977]: random relaxer, random adversary have same expected value vs their worst-case opponents

(We want to lower-bound the outcome for the random relaxer but it's easier to lower-bound the random adversary)

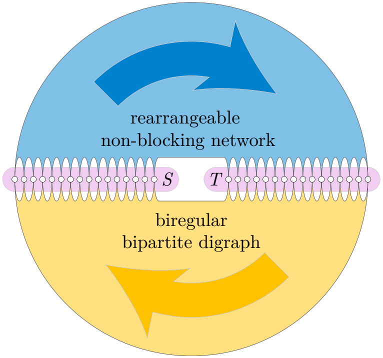
Adversary that picks a Hamiltonian path uniformly at random gets $\approx$ half as much per step as the greedy deterministic adversary

Main idea: Sparse networks

Divide graph into two parts:

- Pool of edges from which adversary chooses far along the relaxation sequence
- Router allowing any sequence of pool edges to be connected into a path

Adversary makes sequence of greedy or random choices from pool edges



Conclusions and open problems

Bellman–Ford is within a constant factor of optimal among non-adaptive relaxation algorithms

For both upper bounds (Bellman–Ford) and our lower bounds, randomized constant factors are smaller than deterministic

But upper and lower bounds are too far apart to prove that optimal deterministic and random constants differ. Do they?

Can we strengthen our model of relaxation algorithms to something more realistic, allowing limited forms of adaptivity?

References and image credits, I

Michael J. Bannister and David Eppstein. Randomized speedup of the Bellman–Ford algorithm. In Conrado Martínez and Hsien-Kuei Hwang, editors, *Proceedings of the 9th Meeting on Analytic Algorithmics and Combinatorics, ANALCO 2012, Kyoto, Japan, January 16, 2012*, pages 41–47. SIAM, 2012. doi: 10.1137/1.9781611973020.6.

J. Barker. Fools gaming. Public domain image, 1785. URL `https://compositor.bham.ac.uk/ornaments/1139846`. From Erasmus's "The praise of folly", translated by W. Kennet, p. 91.

Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 600–611. IEEE, 2022. doi: 10.1109/FOCS54457.2022.00063.

Augustus Binu. Foot Steps at beach. CC-BY-SA image, February 3 2013. URL `https://commons.wikimedia.org/wiki/File:Foot_Steps_1.JPG`.

References and image credits, II

Karl Bringmann, Alejandro Cassis, and Nick Fischer. Negative-weight single-source shortest paths innear-linear time: Now faster! Electronic preprint arxiv:2304.05279, 2023.

United States Marine Corps. Two children compete in a limbo contest at Marine Corps Base Camp Lejeune's Onslow Beach. Public domain image, August 9 2011. URL `https://commons.wikimedia.org/wiki/File:USMC-110808-M-RU378-176.jpg`.

Ramsey W. Haddad and Alejandro A. Schäffer. Recognizing Bellman–Ford-orderable graphs. *SIAM Journal on Discrete Mathematics*, 1(4):447–471, 1988. doi: 10.1137/0401045.

Mark Wheadon. Take Me to Your Leader. CC-BY-SA image, May 3 2008. URL `https://commons.wikimedia.org/wiki/File:Take_Me_to_Your_Leader_(2486871601).jpg`. Image of blue beach binoculars.

References and image credits, III

Andrew Chi-Chih Yao. Probabilistic computations: toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October – 1 November 1977*, pages 222–227. IEEE Computer Society, 1977. doi: 10.1109/SFCS.1977.24.