Title:	Dynamic Approximate All-Pairs Shortest Paths: Breaking the $O(mn)$ Barrier and Derandomization
Name:	Monika Henzinger[1], Sebastian Krinninger[1], Danupon Nanongkai[2]
Affil./Addr. 1:	University of Vienna, Faculty of Computer Science, Austria
Affil./Addr. 2:	Brown University, ICERM, USA
Keywords:	Dynamic graph algorithms; Derandomization; Approximation algorithms; Data structures
SumOriWork:	2013; Henzinger, Krinninger, Nanongkai

Dynamic Approximate All-Pairs Shortest Paths: Breaking the $O(mn)$ Barrier and Derandomization

Monika Henzinger[1], Sebastian Krinninger[1], Danupon Nanongkai[2]

[1] University of Vienna, Faculty of Computer Science, Austria
[2] Brown University, ICERM, USA

Years aud Authors of Summarized Original Work

2013; Henzinger, Krinninger, Nanongkai

Keywords

Dynamic graph algorithms; Derandomization; Approximation algorithms; Data structures

Problem Definition

Given an undirected, unweighted graph with n nodes and m edges that is modified by a sequence of edge insertions and deletions the problem is to maintain a data structure that quickly answers queries that ask for the length $d(u,v)$ of the shortest path between two arbitrary nodes u and v in the graph, called the *distance* of u and v. The fastest *exact* algorithm for this problem is randomized and takes amortized $O(n^2(\log n + \log^2((m+n)/n)))$ time per update and constant query time [6; 11]. In the *decremental* case, i.e., if only edge deletions are allowed, there exists a *deterministic* algorithm with amortized time $O(n^2)$ per deletion [7]. More precisely, its *total update time* for a sequence of up to m deletions is $O(mn^2)$. Additionally there is a randomized algorithm with $O(n^3 \log^2 n)$ total update time and constant query time [1]. However in the decremental case, when only α-*approximate* answers are required, i.e., when it suffices to output an estimate $\delta(u,v)$ such that $d(u,v) \leq \delta(u,v) \leq \alpha d(u,v)$ for all nodes u and v, the total update time can be significantly improved: Let $\epsilon > 0$ be a small constant. The fastest prior work was a class of randomized algorithms with total update time $\tilde{O}(mn)$ for $\alpha = 1 + \epsilon$ [10], $\tilde{O}(n^{5/2+O(1/\sqrt{\log n})})$ for $\alpha = 3 + \epsilon$, and $\tilde{O}(n^{2+1/k+O(1/\sqrt{\log n})})$ for $\alpha = 2k - 1 + \epsilon$ [4].

This leads to the question whether for $\alpha = 1+\epsilon$ (a) a total update time of $o(nm)$ is possible and (b) a deterministic algorithm with total update time $\tilde{O}(nm)$ exists.

As pointed out in [3] and several other places, a *deterministic* algorithm is interesting due to the fact that deterministic algorithms can deal with an *adaptive offline adversary* (the strongest adversary model in online computation [5; 2]) while the randomized algorithms developed so far assume an *oblivious adversary* (the weakest adversary model) where the order of edge deletions must be fixed before an algorithm makes random choices.

Key Results

The paper of Henzinger, Krinninger and Nanongkai [8] presents two algorithms for $\alpha = 1 + \epsilon$. The first one is a deterministic algorithm with total update time $\tilde{O}(mn)$. The second one studies a slightly relaxed version of the problem: Given a constant β, let $\delta(u, v)$ be an (α, β)-*approximation* if $d(u, v) \leq \delta(u, v) \leq \alpha d(u, v) + \beta$ for all nodes u and v. The second algorithm is a randomized algorithm with total update time $\tilde{O}(n^{5/2})$ that can guarantee both a $(1 + \epsilon, 2)$ and a $(2 + \epsilon, 0)$ approximation.

The results build on two prior techniques, namely an exact decremental single-source shortest path data structure [7], called *ES-tree*, and the $(1 + \epsilon, 0)$-approximation algorithm of [10], called *RZ-algorithm*. The RZ-algorithm chooses for all integer i with $1 \leq i \leq \log n$, $\tilde{O}(n/(\epsilon 2^i))$ random nodes as *centers* and maintains an ES-tree up to distance 2^{i+2} for each center. For correctness it exploits the fact that the random choice of centers guarantees the following *invariant (I)*: For every pair of nodes u and v with distance $d(u, v)$ there exists with high probability a center c such that $d(u, c) \leq \epsilon d(u, v)$ and $d(c, v) \leq d(u, v)$ The total update time per center is $O(m2^i)$ resulting in a total update time of $\tilde{O}(mn)$. The deterministic algorithm of [8] derandomizes this algorithm by initially choosing centers fulfilling invariant (I) and after each update (a) greedily generating new centers to guarantee that (I) continues to hold and (b) moving the root of the existing ES-trees. To achieve a running time of $\tilde{O}(mn)$ the algorithm is not allowed to create more than $\tilde{O}(n/(\epsilon 2^i))$ many centers for each i. This condition is fulfilled by dynamically assigning each center a set of $\Omega(2^i)$ vertices such that no vertex is assigned to two centers.

The improved randomized algorithm uses the idea of an *emulator*, a sparser *weighted* graph that approximates the distances of the original graph. Emulators were used for dynamic shortest-paths algorithms before [4]. The challenge when using an emulator is that edge deletions in the original graph might lead to edge deletions, edge insertions, or weight increases in the emulator, requiring in principle the use of a fully dynamic shortest-path algorithm on the emulator. Bernstein and Roditty [4] deal with this challenge by using an emulator where the number of *distance changes* between any two nodes can be bounded. However, the RZ-algorithm requires that the number of distance changes between any two nodes changes at most R times before it exceeds R for any integer R with $1 \leq R \leq n$. As the emulator used by Bernstein and Roditty does not fulfill this property, they cannot run the RZ-algorithm on it. The new algorithm does not construct such an emulator either. Instead it builds an emulator where the error introduced by edge insertions is limited and runs the RZ-algorithm with modified ES-trees, called *monotone ES-trees*, on this emulator. The analysis exploits the fact that the distance between any two nodes in the original graph can only *increase* after an edge deletion. Thus, even if an edge deletion leads to changes in the emulator that *decrease* their distance in the emulator, the corresponding ES-trees do not have to be updated, i.e., the distance of a vertex to its root in the ES-tree *never* decreases. The analysis shows that the error introduced through the use of monotone ES-trees in the RZ-algorithm is small so that the claimed approximation ratio is achieved. However, since the ES-trees are run on the sparse emulator the overall running time is $o(mn)$.

Open Problems

The main open problem is to find a similarly efficient algorithm in the *fully dynamic setting*, where both edge insertions and deletions are allowed. A further open problem is to extend the derandomization technique to the *exact* algorithm of [1].

Another challenge is to obtain similar results for *weighted, directed* graphs. We recently extended some of the above techniques to weighted, directed graphs and presented a randomized algorithm with $\tilde{O}(mn^{0.986})$ total update time for $(1+\epsilon)$-approximate *single-source* shortest paths [9].

Recommended Reading

1. Baswana S, Hariharan R, Sen S (2007) Improved decremental algorithms for maintaining transitive closure and all-pairs shortest paths. Journal of Algorithms 62(2):74–92, announced at STOC, 2002
2. Ben-David S, Borodin A, Karp RM, Tardos G, Wigderson A (1994) On the power of randomization in on-line algorithms. Algorithmica 11(1):2–14, announced at STOC, 1990
3. Bernstein A (2013) Maintaining shortest paths under deletions in weighted directed graphs. In: STOC, pp 725–734
4. Bernstein A, Roditty L (2011) Improved Dynamic Algorithms for Maintaining Approximate Shortest Paths Under Deletions. In: SODA, pp 1355–1365
5. Borodin A, El-Yaniv R (1998) Online computation and competitive analysis. Cambridge University Press
6. Demetrescu C, Italiano GF (2004) A New Approach to Dynamic All Pairs Shortest Paths. Journal of the ACM 51(6):968–992, announced at STOC, 2003
7. Even S, Shiloach Y (1981) An on-line edge-deletion problem. Journal of the ACM 28(1):1–4
8. Henzinger M, Krinninger S, Nanongkai D (2013) Dynamic approximate all-pairs shortest paths: Breaking the $O(mn)$ barrier and derandomization. In: FOCS
9. Henzinger M, Krinninger S, Nanongkai D (2014) Sublinear-Time Decremental Algorithms for Single-Source Reachability and Shortest Paths on Directed Graphs. In: STOC
10. Roditty L, Zwick U (2012) Dynamic Approximate All-Pairs Shortest Paths in Undirected Graphs. SIAM Journal on Computing 41(3):670–683, announced at FOCS, 2004
11. Thorup M (2004) Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In: SWAT, pp 384–396