Deterministic Decremental SSSP and Approximate Min-Cost Flow in Almost-Linear Time

Aaron Bernstein  
Rutgers University New Brunswick  
USA  
bernesti@gmail.com

Maximilian Probst Gutenberg  
ETH Zurich  
Switzerland  
maximilian.probst@outlook.com

Thatchaphol Saranurak  
University of Michigan  
USA  
thsa@umich.edu

Abstract—In the decremental single-source shortest paths problem, the goal is to maintain distances from a fixed source \( s \) to every vertex \( v \) in an \( m \)-edge graph undergoing edge deletions. In this paper, we conclude a long line of research on this problem by showing a near-optimal deterministic data structure that maintains \((1 + \epsilon)\)-approximate distance estimates and runs in \( m^{1+o(1)} \) total update time.

Our result, in particular, removes the oblivious adversary assumption required by the previous breakthrough result by Henzinger et al. [FOCS’14], which leads to our second result: the first almost-linear time algorithm for \((1 - \epsilon)\)-approximate min-cost flow in undirected graphs where capacities and costs can be taken over edges and vertices. Previously, algorithms for max flow with vertex capacities, or min-cost flow with any capacities required super-linear time. Our result essentially completes the picture for approximate flow in undirected graphs.

The key technique of the first result is a novel framework that allows us to treat low-diameter graphs like expanders. This allows us to harness expander properties while bypassing shortcomings of expander decomposition, which almost all previous expander-based algorithms needed to deal with. For the second result, we break the notorious flow-decomposition barrier from the multiplicative-weight-update framework using randomization.

Keywords-component; formatting; style; styling;

I. INTRODUCTION

One of the most fundamental problems in graph algorithms is the single-source shortest paths (SSSP) problem where given a source vertex \( s \) and a undirected, weighted graph \( G = (V, E, w) \) with \( n = \mid V \mid, m = \mid E \mid \), we want to find the shortest paths from \( s \) to every vertex in the graph. This problem has been studied since the 1950s [1], [2] and can be solved in linear time [3].

A natural extension of SSSP is to consider a dynamic graph \( G \) that is changing over time. The most natural model is the fully dynamic one, where edges can be inserted and deleted from \( G \). Unfortunately, recent progress on conditional lower bounds [4], [5], [6] essentially rules out any fully dynamic algorithm with small update and query times for maintaining distances from \( s \). For this reason, most research has focused on the decremental setting, where the graph \( G \) only undergoes edge deletions. In addition to being a natural relaxation of the fully dynamic model, the decremental setting is extremely well-motivated for the SSSP problem in particular: a fast data structure for decremental SSSP can be used as a subroutine within the multiplicative weighted update (MWU) framework to speed up algorithms for various (static) flow problems.

Our main contribution is an almost-optimal data structure for decremental SSSP, which we in turn use to develop the first almost-optimal algorithms for approximate vertex-capacitated max flow and min-cost flow.

A. Previous Work

For our discussion of related work, we assume for \((1 + \epsilon)\)-approximations that \( \epsilon > 0 \) is constant to ease the discussion. We use \( \widetilde{O} \)- and \( \tilde{O} \)-notation to suppress logarithmic and subpolynomial factors in \( n \), respectively.

Decremental Single-Source Shortest Paths (SSSP): A seminal result for decremental SSSP is an algorithm by Even and Shiloach [7] with total update time \( O(mn) \) over the entire sequence of updates in unweighted graphs. Conditional lower bounds indicate that this is near-optimal [8], [4], [5], [6]. But Bernstein and Roditty showed [9] that there exist faster algorithms if one allows for a \((1 + \epsilon)\)-approximation on the distances (and the corresponding shortest paths). This line of research culminated in a breakthrough result by Forster, Henzinger and Nanongkai [10] (see also [11]) who showed how to maintain \((1 + \epsilon)\)-approximate SSSP in total update time \( \tilde{O}(m \cdot \text{polylog}(W)) \), where \( W \) is the maximum weight ratio.

Towards Efficient Adaptive Data Structures: Although it has near-optimal update time, the \( \tilde{O}(m) \) result of [10] suffers from a crucial shortcoming: it is randomized and only works against an oblivious adversary, i.e. an adversary that fixes the entire sequence in advance. For this reason, the result of [10] cannot be used as a black-box data structure, and in particular cannot be incorporated into the MWU framework for flow algorithms mentioned above.

Over the last years, there has been significant effort towards designing adaptive, or even better deterministic, algorithms with comparable update time guarantees [12],...
Max and Min-Cost Flow: Max flow and min-cost flow problems have been studied extensively since the 1950s [18], [19], [20], [21], [22], [23], [24], [25], [26], [27] and can be solved exactly in time $\tilde{O}(m + n^{1.5}) \log^2 (UC)$ [28] and, for unit-capacity graphs, $\tilde{O}(m^{4/3} \log(C))$ [29] where $U$ is the maximum capacity ratio and $C$ is the maximum cost ratio. An extremely recent algorithm [30] makes progress for exact max flow (no costs) in sparse, capacitated graphs: the running time is $O(m^{3/2 - 1/2\log(U)})$. Although enormous effort has been directed towards these fundamental problems, in directed sparse graphs, the fastest algorithms are still far from achieving almost-linear time.

Therefore, an exciting line of work [31], [32], [33], [34], [35], [36] emerged with the goal of obtaining faster approximation algorithms on undirected graphs. This culminated in $\tilde{O}(m \cdot \text{polylog}(U))$-time algorithms for $(1 + \epsilon)$-approximate max flow [33], [34], [36] and $\tilde{O}(m \cdot \text{polylog}(C))$-time algorithms for min-cost flow when all capacities are infinite [37], [38], [39], both of which require only near-linear time.

Limitations of Existing Approaches: Unfortunately, none of the near-linear-time algorithms above handle vertex capacities or can be generalized to min-cost flow with finite capacities. This severely limits the range of applications of these algorithms.

This limitation seems inherent to the existing algorithms. The most successful approach for approximate max flow [33], [34] is based on obtaining fast $n^{o(1)}$-competitive oblivious routing schemes for the $\ell_\infty$-norm (or $\ell_1$-norm in the case of [40]). But for both oblivious routing in vertex-capacitated graphs [41] and min-cost flow oblivious routing [42], [43] there are lower bounds of $\Omega(\sqrt{n})$ for the possible competitiveness. This would lead to an additional polynomial overhead for these algorithms. There are also some alternative approaches to flow problems, but currently they do not lead to almost-linear time algorithms even for regular edge-capacitated max-flow (see e.g. [31], [32], [44]).

Max Flow and Min-Cost Flow via MWU and Decremental SSSP: In order to overcome limitations in the previous approaches, a line of attack emerged that was originally suggested by [45] and was recently reignited by Chuzhoy and Khanna [14]. The idea is that the MWU framework for solving min-cost flow (see e.g. [46], [47]) can be sped up with a fast adaptive decremental SSSP data structure. In [14], Chuzhoy and Khanna obtained promising results via this approach: an algorithm for max flow with vertex capacities only in $\tilde{O}(n^2 \text{polylog}(U))$ time. But this approach currently has two major challenges towards an $\tilde{O}(m)$ time algorithm:

1. Obtaining a fast adaptive decremental SSSP data structure has proven to be an extremely difficult challenge that even considerable effort could not previously resolve [12], [13], [14], [15], [16], [17].
2. Even given such a data structure, the MWU framework is designed to successively route flows along paths from a source $s$ to a sink $t$. But this implies that the flow decomposition barrier applies to the MWU framework, which might have to send flow on $\Omega(m)$ edges over the course of the algorithm (or $\Omega(n^2)$ edges when only vertex capacities are present).

In this article, we overcome both challenges and complete this line of work.

B. Related Results

Dynamic SSR and SSSP in Directed Graphs: While our article focuses on the decremental SSSP problem in undirected graphs, there is also a rich literature for dynamic SSSP in directed graphs and also for the simpler problem of single-source reachability and the related problem of maintaining strongly-connected components.

For fully-dynamic SSR/ SCC, a lower bound by Abboud and Vassilevska Williams [4] shows that one can essentially not hope for faster amortized update time than $O(m)$.

For decremental SSR/ SCC, a long line of research [48], [49], [50], [51], [52], [53] has recently lead to the first near-linear time algorithm [54]. A recent result by Bernstein, Probst Gutenberg and Saranurak has further improved upon the classic $O(mn)$ total update time barrier to $O(nn^{2/3})$ in the deterministic setting [55].

While incremental SSR can be solved straight-forwardly by using a cut-link tree, the incremental SCC problem is not very well-understood. The currently best algorithms [56], [57] obtain total update time $\tilde{O}(\min\{m^{1/2}, n^2\})$. Further improvements to time $\tilde{O}(\min\{m^{3/2}, n^{4/3}\})$ for sparse graphs are possible for the problem of finding the first cycle in the graph [58], [59], the so-called cycle detection problem.

For fully-dynamic SSSP, algebraic techniques are known to lead to algorithms beyond the $\tilde{O}(m)$ amortized update time barrier at the cost of large query times. Sankowski was the first to give such an algorithm [60] which originally only supported distance queries, however, was recently extended to also support path queries [61]. An algorithm that further improves upon the update time/query time trade-off at the cost of an $(1+\epsilon)$-approximation was given by van den Brand and Nanongkai in [62].

The decremental SSSP problem has also received ample attention in directed graphs [7], [50], [51], [63], [64]. The currently best total update time for $(1+\epsilon)$-approximate decremental SSSP is $\tilde{O}(\min\{n^2, mn^{2/3}\}) \log W)$ as given in [64]. Further, [55] can be extended to obtain a deterministic $\tilde{O}(n^{2+2/3} \log W)$ total update time algorithm.

The incremental SSSP problem has also been considered by Probst Gutenberg, Wein and Vassilevska Williams in
[6] where they propose a $\tilde{O}(n^2 \log W)$ total update time algorithm.

**Dynamic APSP.:** There is also an extensive literature for the dynamic all-pairs shortest paths problems.

In the fully-dynamic setting a whole range of algorithms is known for different approximation guarantees, and for the particular setting of obtaining worst-case update times [65], [66], [67], [68], [8], [69], [70], [71], [72], [73], [74], [75], [62], [76]. Most relevant to our work is a randomized $\tilde{O}(m)$ amortized update time algorithm by Bernstein [70] that obtains a $(2 + \epsilon)$-approximation. An algorithm with faster update time is currently only known for very large constant approximation [72].

Similarly, in the decremental setting there has been considerable effort to obtain fast algorithms [77], [9], [78], [10], [74], [79], [80], [15], [17], [81], [82]. We explicitly highlight two contributions for undirected graphs: in [74], the authors obtain a $O(mn \log n)$ deterministic $(1 + \epsilon)$-approximate APSP algorithm (a simpler proof of which can be found in [15]) and in [80] an algorithm is presented that for any positive integer $k$ maintains a $(1 + \epsilon)(2k - 1)$-approximate decremental APSP in time $\tilde{O}(mn^{1/k} \text{polylog} W)$.

The incremental APSP problem has also recently been studied [83].

**Hopsets.:** We also give a brief introduction to the literature on hopsets. Originally, hopsets were defined and used in the parallel setting in seminal work by Cohen [84]. However, due to their fundamental role in both the parallel and the dynamic graph setting, hopsets have remained an active area of development. Following lower bounds on the existential guarantees of hopsets [85], first Elkin and Neiman [86] and then Huang and Pettie [87] obtained almost optimal hopset constructions, where the latter was based on a small modification to the classic Thorup-Zwick emulators/hopset [88].

**C. Our Results**

**Decremental SSSP.:** Our main result is the first deterministic data structure for the decremental SSSP problem in undirected graph with almost-optimal total update time.

[Decremental SSSP] Given an undirected, decremental graph $G = (V, E, w)$, a fixed source vertex $s \in V$, and any $\epsilon > 1/\text{polylog}(n)$, we give a deterministic data structure that maintains a $(1 + \epsilon)$-approximation of the distance from $s$ to every vertex $t \in V$ explicitly in total update time $m^{1+o(1)} \text{polylog} W$. The data structure can further answers queries for an $(1 + \epsilon)$-approximate shortest $s$-to-$t$ path $\pi(s, t)$ in time $|\pi(s, t)|^{o(1)}$.

This result improves upon the state-of-the-art $\tilde{O}(\min\{m, n^2 \text{polylog} W\})$ total update time in the deterministic (or even adaptive) setting and resolves the central open problem in this line of research.

**Mixed-Capacitated Min-Cost Flow.:** Given our new deterministic SSSP data structure, it is rather straight-forward using MWU-based techniques from [47], [46], [17] to obtain unit-capacity min-cost flow in almost-linear time. We are able to generalize these techniques significantly to work for arbitrary vertex and edge capacities.

[Approximate Mixed-Capacitated Min-Cost Flow] For any $\epsilon > 1/\text{polylog}(n)$, consider undirected graph $G = (V, E, c, u)$, where cost function $c$ and capacity function $u$ map each edge and vertex to a non-negative real. Let $s, t \in V$ be source and sink vertices. Then, there is an algorithm that in $m^{1+o(1)} \log \log C$ time returns a feasible flow $f$ that sends a $(1 - \epsilon)$-fraction of the max flow value from $s$ to $t$ with cost at most equal to the min-cost flow. The algorithm runs correctly with high probability.

Our result resolves one of the three key challenges for the max flow/min-cost problem according to a recent survey by Madry [89]. The state-of-the-art for this problem [28] solved the exact version of this problem in directed graphs and hence obtains significantly slower running time $\tilde{O}((m + n^{1.5}) \cdot \text{polylog}(UC))$ which is still super-linear in sparse graphs.

**D. Applications**

Our two main results have implications for a large number of interesting algorithmic problems. See the full version of this paper for more detailed statements and a discussion of how to obtain the results below.

**Applications of Mixed-Capacitated Min-Cost Flow.:**

- Using a reduction of [90], our result for vertex-capacitated flow yields a $O(\log^3(n))$ approximation to sparsest vertex cut in undirected graphs in $\tilde{O}(m)$ times. This is the first almost-linear-time algorithm for the problem with $\text{polylog}(n)$ approximation.

- Combined with another reduction in [91], our result for sparsest vertex cut yields an $O(\log^3(n))$-approximate algorithm for computing tree-width (and the corresponding tree decomposition) in $\tilde{O}(m)$ time. This is again the first almost-linear-time algorithm with $\text{polylog}(n)$ approximation, except for the special cases where the tree-width is itself sub-polynomial [92] or the graph is extremely dense [17]. (See other work on computing tree-width in [93], [94], [95], [96], [97], [91], [95], [96], [98], [14].)

- The above algorithm then leads to improvement for algorithms that relied on computing an efficient tree decomposition. For example, we speed-up the high-accuracy LP solver by Dong, Lee and Ye [99] that is parameterized by treewidth; we reduce the running time to $\tilde{O}(m \cdot \text{tw}(G_A)^2 \log(1/\epsilon))$, improving upon the previous dependency of $\text{tw}(G_A)^4$.

---

2We point out, however, that our dependency on $\epsilon$ is significantly worse than formulated in [89].
Given any graph $G = (V, E)$ (with associated incidence matrix $B$), $\epsilon > 1/\text{polylog}(n)$, a demand vector $\chi \in \mathbb{R}^n$, (super)-linear, convex functions $c_e, c_v : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for each $e \in E$ and $v \in V$. Let $f^*$ be some flow minimizing

$$
\min_{B^T f = \chi} c(f) = \sum_{e \in E} c_e(|f_e|) + \sum_{v \in V} c_v((B^T |f|)_v).
$$

Then, we can compute a $(1 + \epsilon)$ approximate flow $f$ with $c(f) \leq (1 + \epsilon)c(f^*)$ such that $\|B^T f - \chi\|_1 \leq \epsilon \cdot \|\chi\|_1$, i.e. it routes almost all of the demand $\chi$ in almost-linear time. In particular, this is the first almost-linear time algorithm for flow in the weighted $p$-norm $\|W^{-1} f\|_p$ (since we can minimize $\|W^{-1} f\|_p^p$ by $c_e(x) = (x/w_e)^p$).

Applications of Decremental SSSP: There is currently a large gap between the best-known dynamic graph algorithms against oblivious adversaries and adaptive ones. Much of this gap stems from the problem of finding a deterministic counterpart to picking a random source. Plugging in, either our decremental SSSP as a black-box subroutine or our some techniques that we obtain along the way, we obtain various new adaptive algorithms:

- Decremental $(1+\epsilon)$-approximate all-pairs shortest paths (APSP) in total update time $\tilde{O}(mn)$. (Previous adaptive results only worked in unweighted graphs [74], [15].)
- Decremental $\tilde{O}(1)$-approximate APSP with total update time $\tilde{O}(m)$. We note that this result was also obtained in concurrent work by Chuzhoy [100]. Before these results, even in unweighted graphs, all previously adaptive algorithms for decremental APSP (for any approximation) had total update time at least $\Omega(n^2)$ [74], [15], [17], [82]; for weighted graphs they were even slower. Our result is analogous to the oblivious algorithm of Chechik, though she achieves a stronger $O(\log^2(n))$-approximation [80].
- Fully-dynamic $(2 + \epsilon)$ approximate all-pairs shortest paths with $\tilde{O}(m)$ update time, matching the oblivious result of [70].

E. Technical Contributions

From a technical perspective, our dynamic SSSP result in Theorem I-C is by far our more significant contribution. It requires several new ideas, but we would like to highlight one technique in particular that is of independent interest and might have applications far beyond our result:

Key Technique: Converting any Low-Diameter Graph into an Expander: Several recent papers on dynamic graph algorithms start with the observation that many problems are easy to solve if the underlying graph $G$ is an expander, as one can then apply powerful tools such as expander pruning and flow-based expander embeddings. All of these papers then generalize their results to arbitrary graphs by using expander decomposition: they decompose $G$ into expander subgraphs and then apply expander tools separately to each subgraph. Unfortunately, expander decomposition necessarily involves a large number of crossing edges (or separator vertices) that do not belong to any expander subgraph and need to be processed separately. This difficulty has been especially prominent for decremental shortest paths, where expander-based algorithms had previously been unable to achieve near-linear update time [14], [101], [17], [16].

Our key technical contribution is showing how to apply expander-based tools without resorting to expander decomposition. In a nutshell, we show that given any low-diameter graph $G$, one can in almost-linear time compute a capacity $\kappa(v)$ for each vertex such that the total vertex capacity is small and such that the graph $G$ weighted by capacities effectively corresponds to a weighted vertex expander. We can then apply tools such as expander pruning directly to the low-diameter graph $G$. This allows the algorithm to avoid expander decomposition and instead focus on the much simpler task of computing low-diameter subgraphs. We believe that this technique has the potential to play a key role in designing other dynamic algorithms against an adaptive adversary.

Breaking the Flow Decomposition Barrier for MWU: We also briefly mention our technical contribution for the min-cost flow algorithm of Theorem I-C. Plugging our new data structure into the MWU framework is not by itself sufficient, because as discussed above, existing implementations of MWU necessarily encounter the flow decomposition barrier (see for example [45]), as they repeatedly send flow down an entire $s$-$t$ path. We propose a new (randomized) scheme that maintains an estimator of the flow. While previous schemes have used estimators for the weights [102], [103], [104], we are the first to directly maintain only an estimator of the solution, i.e. of the flow itself. This poses various new problems to be considered: a more refined analysis of MWU is needed, a new type of query operation for the decremental SSSP data structure is necessary, and the flow estimator we compute is only a pseudoflow. We succeed in tackling these issues and provide a broad approach that might inspire more fast algorithms via the MWU framework.

ACKNOWLEDGMENT

Aaron Bernstein is supported by NSF CAREER Grant 1942010. Maximilian Probst Gutenberg is supported by a start-up grant of Rasmus Kyng at ETH Zurich. Work was partially done while at the University of Copenhagen where the author was supported by Basic Algorithms Research Copenhagen (BARC), supported by Thorup’s Investigator Grant from the Villum Foundation under Grant No. 16582. The work was partially done while Thatchaphol Saranurak was affiliated at the Toyota Technological Institute at Chicago.
REFERENCES


