Sensitivity and Dynamic Distance Oracles via Generic Matrices and Frobenius Form

Adam Karczmarz^{*} Piotr Sankowski[†]

Abstract

Algebraic techniques have had an important impact on graph algorithms so far. Porting them, e.g., the matrix inverse, into the dynamic regime improved best-known bounds for various dynamic graph problems. In this paper, we develop new algorithms for another cornerstone algebraic primitive, the Frobenius normal form (FNF). We apply our developments to dynamic and fault-tolerant exact distance oracle problems on directed graphs.

For generic matrices A over a finite field accompanied by an FNF, we show (1) an efficient data structure for querying submatrices of the first $k \ge 1$ powers of A, and (2) a near-optimal algorithm updating the FNF explicitly under rank-1 updates.

By representing an unweighted digraph using a generic matrix over a sufficiently large field (obtained by random sampling) and leveraging the developed FNF toolbox, we obtain:

- a conditionally optimal distance sensitivity oracle (DSO) in the case of single-edge or single-vertex failures, providing a partial answer to the open question of [GR21],
- a multiple-failures DSO improving upon the state of the art [vdBS19] wrt. both preprocessing and query time,
- improved dynamic distance oracles in the case of single-edge updates,
- a dynamic distance oracle supporting vertex updates, i.e., changing all edges incident to a single vertex, in $\widetilde{O}(n^2)$ worst-case time and distance queries in $\widetilde{O}(n)$ time.

1 Introduction

Algebraic techniques have had an important impact on graph algorithms so far. For example, the state-of-the-art maximum matching algorithm in dense non-bipartite graphs [Har09, MS04] is of algebraic nature. Porting some fundamental linear-algebraic concepts, like the matrix inverse, into the dynamic regime has led to non-trivial dynamic algorithms for multiple graph problems, such as reachability, shortest paths, maximum matchings [vdBNS19, San04, San05b, San07], and a unified view on iterative optimization methods [vdB21].

In this paper, we consider another cornerstone algebraic primitive, the Frobenius normal form (FNF). Any square matrix A over a field \mathbb{F} is similar to a block-diagonal matrix $F = \text{diag}(C_{f_1}, \ldots, C_{f_k})$ where each C_{f_i} is the companion matrix of a certain monic polynomial $f_i \in \mathbb{F}[x]$ called an *invariant factor* of A. Such a matrix, along with the corresponding similarity transform and its inverse, constitutes a Frobenius normal form of A. Similarly to the Jordan normal form, FNF encodes the characteristic polynomial of a matrix; however, contrary to the Jordan form, computing it does not require finding zeros of this polynomial. Computing an FNF is

^{*}University of Warsaw and IDEAS NCBR, Poland. a.karczmarz@mimuw.edu.pl. Partially supported by the ERC CoG grant TUgbOAT no 772346 and the National Science Centre (NCN) grant no. 2022/47/D/ST6/02184.

[†]University of Warsaw, IDEAS NCBR, and MIM Solutions, Poland. sank@mimuw.edu.pl. Partially supported by the ERC CoG grant TUgbOAT no 772346 and the National Science Centre (NCN) grant no. 2020/37/B/ST6/04179.

a well-studied problem in the symbolic computation community, e.g., [Gie95, Sto01]. Using FNF, [SW19] reproduced¹ the Yuster-Zwick exact distance oracle's bounds [YZ05] (up to polylogarithmic factors), thus providing a proof of concept of the usage of the Frobenius form in graph data structures. However, it seems that the full potential of FNF and other matrix forms in graph algorithms is yet to be fully uncovered.

In this paper, we develop new data structures maintaining and exploiting the Frobenius normal form of a matrix in the generic case. The genericity assumption is a common one in the computer algebra community and says, broadly speaking, that an algorithm should work "almost always", or, for "all but special cases". A specific genericity assumption (that we also use; see, e.g., [JV05]) for matrices may require that the cells of a matrix, seen as indeterminates, do not satisfy some fixed polynomial equation, or in other words, do not lie on some fixed hypersurface of $\mathbb{F}^{n \times n}$. In particular, such an assumption can be used to ensure that A has a single invariant factor, or, equivalently, that the characteristic polynomial p_A of A equals the minimal polynomial μ_A of A. This property implies that A is similar to the companion matrix of the characteristic polynomial of A, i.e., an FNF of A has a particularly simple form. Single-invariant-factor matrices can be themselves considered generic among all matrices over a finite field \mathbb{F} since the fraction of matrices in $\mathbb{F}^{n \times n}$ not having this property is known to be $1/q^3 + O(1/q^4)$ if q is the field size [NP95]. In the following, when talking about generic matrices, we mean matrices with a single invariant factor.

By representing digraphs with generic matrices, we can apply our FNF developments to dynamic and fault-tolerant *exact distance oracle* problems on general dense directed graphs.

Distance oracles in static, fault-tolerant, and dynamic settings. In the distance oracle problem, the goal is to preprocess the input graph G into a data structure supporting arbitrary-pair distance queries. The distance oracle problem has two trivial solutions. First, one could precompute answers to all the $O(n^2)$ possible queries by solving the all-pairs shortest paths (APSP) problem. The other extreme is to not preprocess the graph G at all, and run an s, t-shortest path algorithm (such as Dijkstra's algorithm) from scratch upon a distance query (s,t). The study of distance oracles concentrates on identifying what non-trivial tradeoffs between space, preprocessing time and query time are attainable, possibly under additional assumptions about the graph class of interest, and whether approximate answers are acceptable.

Real-world networks are subject to link/node failures and evolve in time and thus motivate the study of distance oracles in fault-tolerant and dynamic settings.

In the distance sensitivity oracle (DSO) problem, the goal is to preprocess the input graph G = (V, E) so that queries (s, t, F) asking for the length of the shortest s, t path not going through the subset $F \subseteq V \cup E$ of failed edges or vertices are supported. A DSO may also constrain the number of allowed failures, e.g., require that only a single edge or vertex fails. If only at most k failures are supported, we call such a DSO a k-DSO.

In dynamic scenarios, the input graph G is subject to edge set updates and we seek a data structure supporting distance queries interleaved with graph updates. In the *fully dynamic* setting, the data structure is supposed to accept both edge insertions and edge deletions. In the *incremental* (*decremental*, resp.) setting, only edge insertions (deletions, resp.) are accepted. Some dynamic distance oracles accept *single-edge* updates, whereas other allow *vertex updates*, i.e., changing all (possibly $\Theta(n)$) edges incident to a single vertex at once.

¹The paper [SW19] contains a rather significant error, as confirmed by its authors (personal communication). [SW19] mistakenly state that the diameter D (i.e., the largest finite distance) of a digraph G is bounded by the degree of the smallest invariant factor of its adjacency matrix A. Indeed, D is bounded by the degree of the *minimal* polynomial of A, which equals the largest (and not the smallest) invariant factor of A. Nevertheless, the construction of [SW19] remains correct if it happens that the adjacency matrix of A has a single invariant factor.

1.1 State of the art

Static and dynamic computation of Frobenius normal form. For generic matrices (as defined before), finding an FNF is closely related to computing the characteristic polynomial. [Kel85] showed an $\tilde{O}(n^{\omega})$ time² algorithm computing the characteristic polynomial. [Gie95] was the first to obtain an $\tilde{O}(n^{\omega})$ -time algorithm computing an FNF in the general (non-generic) case, whereas [Sto01] gave a deterministic algorithm running within that near-optimal bound. Computing the Frobenius form has also been studied for sparse matrices and, more generally, in the "black box" model where the input matrix can only be accessed via multiplying it by vectors [Ebe00, Vil00]. In particular, a Frobenius form of a generic matrix (with one invariant factor) can be computed in $\tilde{O}(n^2)$ time plus the time needed to perform $\tilde{O}(n)$ black-box matrix-vector multiplications [Ebe00].

[FS11] studied dynamic maintenance of an FNF of a matrix subject to rank-1 updates (i.e., updates of the form $A := A + ab^T$ for given vectors $a, b \in \mathbb{F}^{n \times 1}$, capturing, e.g., row and column updates). They gave a dynamic algorithm with $\tilde{O}(kn^2)$ worst-case update time, where k is the number of invariant factors of A. For the generic case k = 1, the update time is $\tilde{O}(n^2)$. Their algorithm has a significant limitation though. Even for generic matrices, whereas the block-diagonal matrix F (encoding the characteristic polynomial) similar to A is maintained explicitly, an appropriate similarity transform Q and its inverse such that $A = Q^{-1} \cdot F \cdot Q$ is maintained only implicitly. More specifically, the matrices Q and Q^{-1} can only be accessed by multiplying them via vectors in $\tilde{O}(n^2)$ time which makes processing them troublesome.

Static distance oracles. For general weighted digraphs with large edge weights (say, integral and polynomial in n), no non-trivial preprocessing/space/query trade-offs are known.

For digraphs with small integer weights $\{-W, \ldots, W\}$, [YZ05] gave a non-trivial distance oracle with $\tilde{O}(Wn^{\omega})$ preprocessing time, $O(n^2)$ space and $\tilde{O}(n)$ query time. The query time is also significantly smaller than the $\Theta(n^2)$ cost of running breadth-first search on G. The data structure of [YZ05] can also produce an actual shortest path (not just the distance) upon query. Importantly, in the fundamental case of dense unweighted graphs, preprocessing time of [YZ05] is significantly lower than the best-known unweighted APSP bound [Zwi02] of $\tilde{O}(n^{2+\rho})$, where $\rho \approx 0.529$ is a number such that $\omega(1, \rho, 1) = 1 + 2\rho$ and $\omega(a, b, c)$ is such that one can multiply $n^a \times n^b$ and $n^b \times n^c$ matrices in $O(n^{\omega(a,b,c)})$ time. In fact, computing APSP in unweighted directed graphs is conjectured to require $\Theta(n^{2.5})$ time even if $\omega = 2$ [LPW20] and there are compelling reasons to believe that Zwick's algorithm is near-optimal [CWX21].

Distance sensitivity oracles. Whereas the extreme no-preprocessing tradeoff transfers to faulttolerant and dynamic settings with no change, the precompute-all approach requires much more time and space effort simply because there are much more possible queries to serve. Indeed, a trivial solution would require precomputing $\Theta(n^2 \cdot m^f)$ distances if at most f edge failures are to be supported. Despite this, [BK09] showed a 1-DSO for *real-weighted* digraphs with $\tilde{O}(nm)$ preprocessing time, $\tilde{O}(n^2)$ space and constant query time. Note that their preprocessing matches the state-of-the-art $\tilde{O}(nm)$ APSP bound that is conjectured to be optimal for real-weighted digraphs. [DZ17] improved the space bound of [BK09] to $O(n^2)$. For the case f = 2, [DP09] gave a DSO with $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time, requiring higher polynomial preprocessing.

There is also extensive prior work on distance sensitivity oracles in the unweighted and smallinteger-weights regimes. [WY13] showed the first non-trivial distance sensitivity oracle in this

²Where $\omega \approx 2.37$ is the matrix multiplication exponent, i.e., a number such that one can multiply two $n \times n$ matrices in $O(n^{\omega})$ time.

setting. For integer weights $\{-W, \ldots, W\}$, they could achieve subcubic preprocessing and subquadratic query time for any constant number of failures. In the same regime, [GW20a] showed the first 1-DSO with subcubic preprocessing and *sublinear* query time. [CC20] showed a 1-DSO with $O(Wn^{2.873})$ preprocessing and *polylogarithmic* query time. Around the same time, [Ren22] gave a 1-DSO for *positive* edge weights with $O(Wn^{2.724})$ preprocessing time and O(1) query time. The data structures [CC20, GW20a, Ren22, WY13] are all randomized, have a linear dependence on the largest (absolute) edge weight, and also support path reporting. [BCC⁺22] showed a derandomization of the approach of [Ren22] at the cost of slightly slower (but still subcubic) preprocessing.

The aforementioned data structures for small weights all leverage fast matrix multiplication to speed up combinatorial computations. DSOs with improved preprocessing and query times have been obtained via a more aggressive use of algebraic techniques: forms of path counting [DI05, KS02] or small-rank update to the matrix inverse [San04, San05b, San05a] combined with randomized polynomial identity testing [Zip79]. These techniques typically do not allow for efficient path reporting. Using algebraic techniques of this flavor, [GR21] recently showed a 1-DSO for digraphs with weights $\{1, \ldots, W\}$ with $O(Wn^{2.58})$ preprocessing time that is quite close to the $O(n^{2.529})$ APSP bound of [Zwi02].

[vdBS19] gave an algebraic DSO that can handle a *polynomial* number of failures in the case of weights $\{-W, \ldots, W\}$. Specifically, for any $\mu \in [0, 1]$, their data structure has $\widetilde{O}(Wn^{\omega+(3-\omega)\mu})$ construction time, and after preprocessing a batch of f failures in $\widetilde{O}(Wn^{2-\mu}f^2 + Wnf^{\omega})$ time, answers distance queries wrt. that batch in $\widetilde{O}(Wn^{2-\mu}f + Wnf^2)$ time. That is, if the failing edges are considered a part of the query, the query time is $\widetilde{O}(Wn^{2-\mu}f^2 + Wnf^{\omega})$. In particular, one can handle up to $f = n^{1/\omega-\epsilon} \approx n^{0.42}$ failures with subcubic preprocessing and subquadratic query time.

Fully dynamic exact distance oracles. For general weighted digraphs, [DI04] gave a combinatorial deterministic fully dynamic data structure (later slightly improved by [Tho04]) with $\tilde{O}(n^2)$ amortized update time maintaining all-pairs shortest paths explicitly. This improves upon recompute-from-scratch for all but the sparsest digraphs. The fully dynamic APSP problem (that is, explicitly maintaining the distance matrix) has also been studied with the objective of optimizing the *worst-case* update bounds [ACK17, CZ, GW20b, Tho05]. The current best-known worst-case update bound for APSP is $\tilde{O}(n^{2+2/3})$ for weighted graphs and $\tilde{O}(n^{2.5})$ for unweighted graphs [ACK17, GW20b]. In particular, the latter improves upon the static APSP bound of [Zwi02]. Interestingly, all the known fully dynamic APSP data structures support vertex updates.

As far as fully dynamic exact distance oracles with a non-trivial query procedure are concerned, [RZ11] showed a data structure tailored to sparse graphs with $\tilde{O}(m\sqrt{n})$ amortized (vertex) update time and $\tilde{O}(n^{3/4})$ query time, whereas [KS23] recently presented a data structure for real-weighted digraphs with $\tilde{O}(mn^{4/5})$ worst-case update time and $\tilde{O}(n^{4/5})$ query time.

For dense graphs, dynamic distance oracles with both subquadratic *single-edge* update and query time can be obtained using variants of dynamic matrix inverse [San04, vdBNS19]. The state-of-the-art worst-case update/query bound of this kind is $\tilde{O}(n^{1.703})$ due to [vdBFN22]. Moreover, [AvdB23, BHG⁺21] described shortest path-reporting extensions of these algebraic data structures with polynomially worse (but still subquadratic) worst-case update and query time.

1.2 Our results

Frobenius form toolbox. We obtain two tools for generic matrices (i.e., with a single invariant factor) over an arbitrary finite field \mathbb{F} and accompanied by a Frobenius form. The first one is a data structure for querying some number of initial powers of the matrix.

Theorem 1.1. Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix and suppose its Frobenius normal form is given. One can preprocess A in $\widetilde{O}(n^2)$ time so that the following queries are supported.

Given $S, T \subseteq [n]$ and $h \in [1, n]$, compute the $S \times T$ submatrices of the matrix powers A^1, \ldots, A^h . The query time is $\widetilde{O}(n^{\omega(s, 1-\alpha, t)+\alpha})$, where $|S| = \lfloor n^s \rfloor$, $|T| = \lfloor n^t \rfloor$ and $h = \lfloor n^{\alpha} \rfloor$.

Theorem 1.1 generalizes and improves upon a previous result of [SW19] who showed³ that after additional $\tilde{O}(n^{\omega})$ -time preprocessing of a generic A (given its Frobenius form), one can support queries (i, j) asking for the n values $(A^1)_{i,j}, (A^2)_{i,j}, \ldots, (A^n)_{i,j}$ in $\tilde{O}(n)$ time.

One particularly important use case of Theorem 1.1 is computing the first $h \leq n^{\alpha}$ powers of A. Theorem 1.1 implies that this is possible in $\tilde{O}(hn^{\omega(1,1-\alpha,1)})$ time which polynomially improves upon the trivial $O(hn^{\omega})$ bound for all polynomial values of h. To the best of our knowledge, previously, the first non-trivial result of this kind has been described for the case h = n: [Sto15, ZLS15] showed that n initial powers of A can be computed in $\tilde{O}(n^3)$ time, which also follows from the data structure of [SW19]. An improved $\tilde{O}(h^2n^{\omega(1,1-\alpha,1-\alpha)})$ bound for computing the initial h powers of A has been shown (implicitly) by [GR21]. Both [Sto15, ZLS15] and [GR21] studied a more general problem of inverting an arbitrary degree-d polynomial matrix modulo x^{h+1} .

We also show an improved dynamic algorithm updating a Frobenius form of a generic matrix explicitly subject to a rank-1 perturbation.

Theorem 1.2. Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix. Suppose an FNF of A and an FNF of A^T are given. Then, for any $a, b \in \mathbb{F}^{n \times 1}$ such that $A' = A + ab^T$ is generic, Frobenius normal forms of A' and $(A')^T$ can be computed explicitly in $\widetilde{O}(n^2)$ time. The algorithm succeeds with high probability.

Here, the assumption that an FNF of the transpose is also given is without much loss of generality. Indeed, in a typical scenario, some FNF of A is initialized in, say, $\tilde{O}(n^{\omega})$ time before the first application of Theorem 1.2. If we additionally compute an FNF of A^T at that point within the same asymptotic bound, every subsequent application of Theorem 1.2 updates both FNFs.

The crucial advantage of Theorem 1.2 compared to the dynamic algorithm of [FS11] is that the FNFs – including the (inverse) similarity transforms – are updated explicitly. This property is essential if we want to use Theorem 1.2 in combination with the data structure of Theorem 1.1 which requires an explicit FNF of the input matrix.

Applications to distance oracles. As an application of the developed tools for generic matrices with a Frobenius normal form, we show improved algebraic distance sensitivity oracles and fully dynamic distance oracles for *unweighted* dense directed graphs.

First of all, we show that for the 1-DSO problem, one can essentially match the static APSP bound of [Zwi02] that is conjectured to be near-optimal [CWX21, LPW20].

Theorem 1.3. Let G be an unweighted digraph. In $\tilde{O}(n^{2+\rho}) = O(n^{2.529})$ time one can construct a distance sensitivity oracle for G handling single-edge/vertex failures with O(1) query time and $\tilde{O}(n^2)$ space. The data structure is Monte Carlo randomized and the produced answers are correct with high probability⁴.

[GR21] asked whether a 1-DSO with preprocessing time $\tilde{O}(Wn^{2+\rho})$ is possible for graphs with weights $\{1, \ldots, W\}$. Theorem 1.3 yields an affirmative answer to this problem in the case W = 1. For distance oracles handling many failures, we show:

³This result does not depend on the erroneous statement in [SW19] about the graph diameter.

⁴That is, with probability at least $1 - 1/n^c$, where the constant $c \ge 1$ can be set arbitrarily. We will also use the standard abbreviation w.h.p.

Theorem 1.4. Let G be an unweighted digraph. There exists a distance sensitivity oracle with $\widetilde{O}(n^{\omega})$ preprocessing and $O(n^2)$ space such that for any set F of f edge or vertex failures, the data structure can be updated in $\widetilde{O}(nf^{\omega-1})$ time to support distance queries with failures F in $\widetilde{O}(nf)$ time. The data structure is Monte Carlo randomized and the produced answers are correct w.h.p.

For unweighted digraphs, the data structure of Theorem 1.4 improves upon the state-of-theart [vdBS19] in terms of update and query time even if [vdBS19] uses cubic preprocessing (i.e., if one sets $\mu = 1$). In particular, if the failures are part of the query, then we can handle a distance query under up to $n^{1/(\omega-1)-\epsilon} \approx n^{0.72}$ failures in subquadratic time. Moreover, in the case of $f = \tilde{O}(1)$ failures, our data structure has preprocessing and query time matching the respective time characteristics of the state-of-the-art (failure-free) distance oracle of [YZ05].

That being said, the distance oracles of Theorems 1.3 and 1.4 have some evident drawbacks compared to the respective results of [GR21, vdBS19]. Our data structure can be generalized to handle small positive weights [1, W] at the cost of introducing a multiplicative factor polynomial in W – by replacing n with nW in the preprocessing and query bounds. That is, the respective preprocessing times in Theorems 1.3 and 1.4 for weighted graphs should be replaced with $\tilde{O}((Wn)^{2.529})$ and $\tilde{O}((Wn)^{\omega})$, respectively. On the other hand, the previously known data structures achieve a linear dependence on W. Additionally, our data structure of Theorem 1.4 does not seem to generalize to negative edge weights, whereas that of [vdBS19] does.

Let us now move to our results in the dynamic scenario. First, we obtain improved bounds for fully dynamic distance oracles supporting single-edge updates in unweighted digraphs.

Theorem 1.5. Let G be an unweighted digraph. There exists a Monte Carlo randomized data structure maintaining G under single-edge insertions and deletions and supporting s,t-distance queries with $O(n^{1.673})$ worst-case update and query time. The answers produced are correct w.h.p.

Theorem 1.5 is obtained via a small tweak to the data structure of [vdBFN22] using Theorem 1.1. Interestingly, if $\omega = 2$, the update/query bounds of both data structures (Theorem 1.5 and that of [vdBFN22]) simplify to an odd-looking bound of $\tilde{O}(n^{1+5/8}) = \tilde{O}(n^{1.625})$.

One component of [vdBFN22] is periodically recomputing bounded-hop all-pairs distances. Using a different approach avoiding this entirely, we obtain another dynamic distance oracle.

Theorem 1.6. Let G be an unweighted digraph. There exists a Monte Carlo randomized data structure maintaining G under single-edge insertions and deletions and supporting s,t-distance queries with $\widetilde{O}\left(n^{\frac{\omega+1}{2}}\right)$ worst-case update and query time. The answers produced are correct w.h.p.

The update/query bound of $\tilde{O}(n^{(\omega+1)/2}) = O(n^{1.687})$ is currently inferior to that of Theorem 1.5 but nevertheless superior to the state-of-the-art bound $O(n^{1.703})$ [vdBFN22]. However, the data structure of Theorem 1.6 might be considered more promising: if $\omega = 2$, its update bound simplifies to a natural $\tilde{O}(n^{1.5})$ bound, and even if $\omega < 2.25$, the $\tilde{O}(n^{(\omega+1)/2})$ bound is better than the theoretical limit of the approach of [vdBFN22].

Finally, we achieve a very natural tradeoff in the more general case of vertex updates.

Theorem 1.7. Let G be an unweighted digraph. There exists a Monte Carlo randomized data structure maintaining G under fully dynamic vertex updates in $\tilde{O}(n^2)$ worst-case time per update and supporting arbitrary pair distance queries in $\tilde{O}(n)$ time. The answers are correct w.h.p.

Theorem 1.7 shows that one can preserve the linear query of the static distance oracle of [YZ05] without rebuilding it from scratch. Similarly as in the case of the previously known data structures supporting vertex updates for transitive closure [San04] and APSP, our data structure does not need

fast matrix multiplication for performing updates or queries within the stated bounds. Another interesting consequence of Theorem 1.7 is that we can maintain distances between *n* arbitrary pairs of vertices in $\tilde{O}(n^2)$ worst-case time per update. To the best of our knowledge, no previous data structure could achieve that for unweighted dense graphs. [Kar21] showed that $\tilde{O}(mn^{2/3})$ worst-case update time is possible for sparse weighted digraphs if the pairs of interest are fixed.

Similarly as for distance oracles, our dynamic data structures generalize to digraphs with weights $\{1, \ldots, W\}$ at the cost of a $o(W^2)$ factor in the respective bounds. Again, the previous best bound [vdBFN22] can be easily lifted to this case with an overhead linear in W.

1.3 Technical overview

Generic matrices and Frobenius form. Let $U \cdot C \cdot U^{-1}$ be a Frobenius form of a generic matrix $A \in \mathbb{F}^{n \times n}$, where $C \in \mathbb{F}^{n \times n}$ is the companion matrix of the characteristic polynomial of A, and $U \in \mathbb{F}^{n \times n}$ is a (not necessarily unique) invertible similarity transform. See Section 3 for more detailed definitions of these notions. As observed by [SW19], since C is a companion matrix:

- (1) the *n* matrices $U \cdot C, U \cdot C^2, \ldots, U \cdot C^n$ can be computed in $\widetilde{O}(n^{\omega})$ time and stored explicitly (albeit succinctly) in $O(n^2)$ space;
- (2) storing these matrices enables computing the values $A_{i,j}, (A^2)_{i,j}, \ldots, (A^n)_{i,j}$ for any query pair (i,j) in $\widetilde{O}(n)$ time via Hankel matrix-vector multiplication which in turn is easily reducible to polynomial multiplication, i.e., FFT [CT65]. See, e.g., [GVL13].

To obtain Theorem 1.1, we first show a more efficient $\widetilde{O}(n^2)$ -time algorithm for computing the matrices $U \cdot C, \ldots, U \cdot C^n$ by interpreting this problem as generation of multiple terms of linear recurrences of order n and employing an efficient recent algorithm for this task [BM21].

Next, we generalize item (2) above to queries about an arbitrary submatrix of some $h \leq n$ initial powers. While for small values of h, say $h = \sqrt{n}$, we cannot evaluate the cells (i, j) of h initial powers faster than in $\widetilde{O}(n)$ time, we observe that considerable computational savings (on average) are possible if a larger submatrix $S \times T$ (potentially the full $n \times n$ submatrix), for $S, T \subseteq [n]$, is queried. Indeed, in such a case, a careful packing of the matrices $U \cdot C, U \cdot C^2, \ldots, U \cdot C^n$ into two degree- $\Theta(h)$ polynomial matrices of sizes $|S| \times \lceil n/h \rceil$ and $\lceil n/h \rceil \times |T|$ (resp.) allows us to benefit from FFT [CT65] and fast rectangular matrix multiplication [GU18, HP98] at the same time.

Dynamic Frobenius form. To obtain the improved dynamic Frobenius form data structure of Theorem 1.2, more ideas are needed. Crucially, we use a notion of a generic vector wrt. A, i.e., a vector $u \in \mathbb{F}^{n \times 1}$ such that the *iterates* $u, Au, \ldots, A^{n-1}u$ are linearly independent. In other words, u is generic wrt. A if the order-n Krylov subspace generated by A and u has dimension n. As shown by [Kel85], if u is generic wrt. A, the iterates of u encode a similarity transform U such that $U \cdot C \cdot U^{-1}$ is a Frobenius form of A (and C is a companion matrix of p_A). Using the techniques of [BGY80, Ebe00], one can in fact prove that, given the iterates of a generic vector u wrt. A and iterates of a generic vector v wrt. A^T , all three matrices U, C, U^{-1} comprising the Frobenius form can be computed in $\tilde{O}(n^2)$ time. Moreover, that algorithm can be used to detect non-genericity of input vectors: if either u or v is not generic wrt. the respective matrix, the algorithm fails. Therefore, if A is subject to a rank-1 update $A' := A + ab^T$ (which keeps A' generic), then in order to compute a Frobenius form of A' explicitly in $\tilde{O}(n^2)$ time, it is sufficient to compute the iterates of some generic vectors u', v' wrt. the matrices $A + ab^T$ and $(A + ab^T)^T$ in $\tilde{O}(n^2)$ time.

We show a dynamic programming-based algorithm for this task that works even if u', v' are not generic. Specifically, given FNFs of generic A and A^T and some arbitrary vectors $u', v' \in \mathbb{F}^{n \times 1}$,

the algorithm computes the iterates of u' wrt. $A + ab^T$ and v' wrt. $(A + ab^T)^T$. One important ingredient here is the preprocessing of Theorem 1.1 which also allows computing iterates of an arbitrary vector in near-optimal $\tilde{O}(n^2)$ time. With the respective iterates wrt. A and A^T in hand, the obtained dynamic programming formula for the subsequent iterates wrt. the perturbed matrices can be efficiently evaluated using a folklore combination of divide-and-conquer and FFT.

Finally, as proved by [BGL03], generic vectors wrt. a generic $n \times n$ matrix over a finite field can be obtained (w.h.p.) within $\tilde{O}(\text{polylog } n)$ random samples even for small fields. Thus, computing the iterates of O(polylog n) random vectors wrt. $A + ab^T$ and $(A + ab^T)^T$ and feeding them into the aforementioned procedure based on [BGY80, Ebe00] yields an $\tilde{O}(n^2)$ -time Las Vegas randomized (w.h.p.) algorithm computing an FNF of a (generic) matrix A after a rank-1 perturbation.

From generic matrices to graphs. The key technical idea enabling all our developments for fault-tolerant and dynamic distance oracles is to represent an *arbitrary* directed graph using a *generic matrix*.

Roughly speaking, the state-of-the-art dynamic distance oracle [vdBFN22] for unweighted graphs (and its predecessors [vdBNS19, San05b]) rely on *path counting* of sufficiently short paths. If A(G)is an adjacency graph of G, then $(A(G)^k)_{s,t}$ equals the number of distinct $s \to t$ paths consisting of precisely k edges. As a result, if $(A(G)^k)_{s,t} \neq 0$, the distance from s to t is no more than k. In the other direction, if an $s \to t$ path of length k exists in G, then $(A(G)^k)_{s,t} \neq 0$. Consequently, the matrix powers $A(G), A(G)^2, \ldots, A(G)^h$ encode the *short* distances between vertices at distance at most h. The short distances (for a sublinear h = poly(n)), combined with standard hitting set arguments [UY91], already allow computing an s, t distance in G in subquadratic time.

The challenge is to efficiently compute the first h matrix powers of A(G) and maintain them (possibly implicitly) under element updates to A(G). The simple-minded approach leads to an $\tilde{O}(n^{\omega} \cdot h^2)$ time for the static computation since the elements of $A(G)^h$ may use up to $\tilde{O}(h)$ bits. In the path applications, we are only interested in whether the entries of the powers are zero or not, so performing all the counting modulo a sufficiently large random $\tilde{O}(\text{polylog } n)$ -bit prime still yields high-probability correctness. Therefore, the powers $A(G), \ldots, A(G)^h$ can be thought to be computable in $\tilde{O}(n^{\omega} \cdot h)$ time. The well-established way to handle updates is to note that these powers are encoded by the inverse of the polynomial matrix $I - A(G) \cdot X$ in the ring of polynomials modulo X^{h+1} and apply dynamic matrix inverse data structures [San04, vdBNS19, vdBFN22]. Since the entries of $A(G)^h$ are polynomials of degree at most h, the obtained update bounds are generally factor- $\tilde{O}(h)$ away from the known dynamic matrix inverse bounds [San04, vdBNS19].

As shown by [Sto15, ZLS15], the $O(n^{\omega} \cdot h)$ bound for computing the first h powers statically is certainly not optimal for large values of h: for h = n the computation can be performed in nearoptimal $\tilde{O}(n^3)$ time precisely via a reduction to polynomial matrix inverse. In fact, the state of the art DSOs [vdBS19, GR21] that we improve upon rely on the techniques of [ZLS15]. Specifically, one of the contributions of [GR21] is showing that $h = n^{\alpha}$ first powers of a matrix can be computed in $\tilde{O}(n^{\omega(1,1-\alpha,1-\alpha)+2\alpha})$ field operations.

To obtain the improved distance oracles, we avoid using the techniques of [ZLS15] and apply our data structure for querying matrix powers (Theorem 1.1) to compute the *h*-bounded distances faster, in $\tilde{O}(hn^{\omega(1,1-\alpha,1)})$ time (after $\tilde{O}(n^{\omega})$ -time computation of an FNF). Such a speed-up alone is enough to obtain the tweaked dynamic distance oracle of Theorem 1.5, and, via the reduction of [GR21], a 1-DSO with preprocessing time matching the APSP bound of [Zwi02].

For the above application of Theorem 1.1 to be legitimate, we need to guarantee that the graph is represented by a generic matrix. Simply using a standard adjacency matrix of a digraph fails here, since the adjacency matrix is often not generic, i.e., it may have multiple invariant factors. Based on the techniques of [Wie86], we show that a weighted adjacency matrix $A(G) \in \mathbb{F}^{n \times n}$ representing G can be appropriately and efficiently sampled so that A(G) is generic, if the size of the field \mathbb{F} used is sufficiently large but still polynomial in n. A random weighted adjacency matrix also has, with high probability, the desired properties relating the non-zero entries of $(A(G))^k$ to the existence of length- $\leq k$ paths between vertex pairs. Moreover, the elements of A(G) are sampled independently and thus small updates to the graph G result in small updates to A(G) and they maintain genericity with high probability (over polynomially many updates).

A further combination of the submatrix queries data structure of Theorem 1.1 with formulas for updating the matrix inverse after changing few elements [vdBNS19, San04, San05b] and standard hitting set arguments [UY91] allows us to obtain the improved multiple-failures distance sensitivity oracle of Theorem 1.4 and the "prospective" dynamic distance oracle of Theorem 1.6.

We note that the previous work, in particular concerning (possibly negatively-) weighted graphs (e.g., [vdBS19, San05a]) or reachability in presence of cycles [San04], where simple path counting fails, also used weighted adjacency matrices, albeit for a different reason. There, one starts with a polynomial *symbolic adjacency matrix* in the first place and then applies random variable substitution to enable efficient polynomial identity testing [Sch80, Zip79]. Our use of weighted adjacency matrices can be considered merely a trick to fix the non-genericity in basic path counting.

1.4 Further related work

Specialized exact distance oracles have been shown for incremental [AIMN91] and decremental [BHS07, EFGW21] unweighted directed graphs. Non-trivial *approximate* distance oracles for weighted directed graphs are known in the fully dynamic setting [vdBN19] and partially dynamic settings [Ber16, EFGW21, KŁ19].

There has been extensive and influential work on static distance oracles for *undirected graphs*, especially in various approximate settings, e.g., [Che15, CZ22, PR14, TZ05, Wul12]. See also the survey [Som14]. Distance oracles for undirected graphs have also been studied specifically in the fault-tolerant (e.g., [CLPR12, DR22]) and fully dynamic (e.g., [Ber09, vdBN19]) settings.

2 Preliminaries

We denote by [n] the set $\{1, \ldots, n\}$. Let \mathbb{F} be a finite field. For an $n \times m$ matrix $A \in \mathbb{F}^{n \times m}$, and $S \subseteq [n], T \subseteq [m]$, we generally denote by $A_{S,T}$ the submatrix of A with rows S and columns T. We may write $A_{s,T}$ or $A_{S,t}$, for $s, t \in [n]$, to denote $A_{\{s\},T}$ or $A_{S,\{t\}}$, respectively. In particular, $A_{s,t}$ is the element in the cell (s,t) of A. Whenever we write A_S , we mean $A_{S,S}$.

If v is a column (row) vector in $\mathbb{F}^{n\times 1}$ (in $\mathbb{F}^{1\times n}$, resp.), then we sometimes write v_i to denote $v_{i,1}$ ($v_{1,i}$, resp.). If n is known from the context, we denote by $e_i \in \mathbb{F}^{n\times 1}$ a column vector satisfying $(e_i)_j = [j = 1]$.

We generally measure time in field operations, i.e., the field operations are assumed to take unit time. We denote by MM(p,q,m) the time needed to multiply a matrix from $\mathbb{F}^{p\times q}$ by a matrix from $\mathbb{F}^{q\times m}$. That is, if $p = |n^{\alpha}|, q = |n^{\beta}|, m = |n^{\gamma}|$ for $\alpha, \beta, \gamma \geq 0$, then $MM(p,q,m) = O(n^{\omega(\alpha,\beta,\gamma)})$.

When talking about directed graphs G = (V, E), for $F \subseteq E \sqcup V$ we denote by G - F the graph obtained from G by removing the vertices and/or edges F. If G is (non-negatively) weighted, then we denote by $w_G(uv)$ the weight of the edge $uv \in E$. For any $s, t \in V$, denote by $\delta_G(s, t)$ the weight of the shortest $s \to t$ path in G. If no $s \to t$ path exists in G, we put $\delta_G(s, t) = \infty$.

3 Generic matrices and Frobenius form

Let \mathbb{F} be a finite field. Once again, we call a matrix $A \in \mathbb{F}^{n \times n}$ generic if the characteristic polynomial $p_A(t) = \det(tI - A)$ equals the minimal polynomial μ_A of A, i.e., the minimum-degree monic polynomial over \mathbb{F} such that $\mu_A(A) \equiv 0$. We start with the following well-known fact.

Fact 3.1. (see, e.g., [BJN94]) Suppose the matrix A is generic and let $p_A(t) = t^n + c_{n-1}t^{n-1} + \ldots + c_0$. There exists an invertible matrix $U \in \mathbb{F}^{n \times n}$ such that

$$A = U \cdot C \cdot U^{-1},\tag{1}$$

where $C \in \mathbb{F}^{n \times n}$ is a companion matrix of p_A , that is:

$$C = \begin{bmatrix} 0 & 0 & \dots & 0 & -c_0 \\ 1 & 0 & \dots & 0 & -c_1 \\ 0 & 1 & \dots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & 1 & -c_{n-1} \end{bmatrix}.$$

For any similarity transform $U \in \mathbb{F}^{n \times n}$ satisfying Equation (1), $U \cdot C \cdot U^{-1}$ is called the Frobenius normal form (FNF) of the generic matrix A. The Frobenius form can be defined more generally for arbitrary non-generic matrices from $\mathbb{F}^{n \times n}$, albeit the middle matrix C has then a more complicated form – it may consist of multiple companion matrices. FNF can be computed deterministically in $\tilde{O}(n^{\omega})$ [Sto01] time for every matrix from $\mathbb{F}^{n \times n}$. For generic matrices, there exists an easier $\tilde{O}(n^{\omega})$ -time FNF algorithm [Kel85] that we sketch in this section and build upon later on.

Let us call a vector $u \in \mathbb{F}^{n \times 1}$ generic wrt. A if the vectors $u, Au, A^2u, \ldots, A^{n-1}u$ are linearly independent.

Lemma 3.2. [Kel85] Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix and let $u \in \mathbb{F}^{n \times 1}$ be generic wrt. A. Then the matrix $U = \begin{bmatrix} u & Au & A^2u & \dots & A^{n-1}u \end{bmatrix} \in \mathbb{F}^{n \times n}$ is invertible and $U \cdot C \cdot U^{-1}$ is an FNF of A.

Proof. U is invertible since it is of size $n \times n$ and its columns are linearly independent. Moreover, we have $AU = [Au|A^2u|...|A^nu]$. On the other hand, for any vector $v \in \mathbb{F}^{1 \times n}$ such that $v = (v_0, \ldots, v_{n-1})$, we have $vC = (v_1, \ldots, v_{n-1}, -\sum_{i=0}^{n-1} c_i v_i)$. As a result:

$$UC = \begin{bmatrix} Au & A^2u & \dots & A^{n-1}u & -\left(\sum_{i=0}^{n-1} c_i A^i\right)u \end{bmatrix}.$$

Since A is generic, $\mu_A(A) = p_A(A) = A^n + \sum_{i=1}^{n-1} c_i A^i \in \mathbb{F}^{n \times n}$ is a zero matrix. Consequently, we have $UC = \begin{bmatrix} Au & A^2u & \dots & A^nu \end{bmatrix}$, which proves AU = UC. Thus, indeed $A = UCU^{-1}$.

We now refer to [BGL03] for the following estimate.

Theorem 3.3. [BGL03, Theorem 9] Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix and let $q = |\mathbb{F}|$. Then, with probability at least $\frac{0.2}{1 + \log_q n}$, a random vector from $\mathbb{F}^{n \times 1}$ is generic wrt. A.

Since one can compute the *n* vectors $u, Au, \ldots, A^{n-1}u$ and matrix inverse in general in $\widetilde{O}(n^{\omega})$ time (see, e.g., [Kel85]), by Theorem 3.3, Lemma 3.2 applied to random vectors $u \in \mathbb{F}^{n \times 1}$ yields:

Lemma 3.4. Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix. There is a Las Vegas algorithm computing an *FNF* of A in $\widetilde{O}(n^{\omega})$ time. The running time bound holds with high probability.

Proof. Multiplying a random vector $u \in \mathbb{F}^{n \times 1}$ by the *n* first powers of *A* yields an FNF of *A* with probability $\Omega(1/\log n)$. After $c \cdot \log^2 n = O(\operatorname{polylog} n)$ trials (where c = O(1)), the success probability is at least $1 - O(n^{-c})$.

Crucially for our applications, an FNF can be computed faster if an efficient way of multiplying a vector by n powers of A is available. The following lemma has been proven in a more general form by Eberly [Ebe00]. We include a proof for completeness.

Lemma 3.5. [Ebe00] Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix, and let $u, v \in \mathbb{F}^{n \times 1}$. Suppose the vectors $u, Au, \ldots, A^{n-1}u$ and $v^T, v^T A, \ldots, v^T A^{n-1}$ are given.

Then, using $O(n^2)$ additional field operations one can either compute a Frobenius normal form of A or detect that either u is not generic wrt. A or v is not generic wrt. A^T .

Proof. We wish to compute the Frobenius form $U \cdot C \cdot U^{-1}$ given by Lemma 3.2 or detect that either u is not generic wrt. A or v is not generic wrt. A^T . Recall that the matrix U is obtained by putting the (given) vectors $u, Au, \ldots, A^{n-1}u$ in a row. Moreover, let

$$V = \begin{bmatrix} v^T \\ v^T A \\ \vdots \\ v^T A^{n-1} \end{bmatrix}.$$

Consider a Hankel matrix (i.e., with all skew diagonals constant)

$$VU = \begin{bmatrix} v^{T}u & v^{T}Au & \dots & v^{T}A^{n-1}u \\ v^{T}Au & v^{T}A^{2}u & \dots & v^{T}A^{n}u \\ \vdots & \vdots & \ddots & \vdots \\ v^{T}A^{n-1}u & v^{T}A^{n}u & \dots & v^{T}A^{2n-2}u \end{bmatrix}$$

Note that the 2n - 1 distinct entries of VU can be computed using O(n) inner products of the input vectors, i.e., in $O(n^2)$ time. One can check in $\tilde{O}(n)$ time whether an $n \times n$ Hankel matrix (given its O(n) skew diagonal values) is singular [BGY80]. If VU is singular, then either V or U is singular, that is, either v is not generic wrt. A^T or u is not generic wrt. A.

Suppose both U and V are non-singular. We have

$$U^{-1} = (VU)^{-1} \cdot V,$$

so the columns of U^{-1} can be found by solving *n* linear systems Hx = b, where H = VU is a non-singular Hankel matrix and *b* is a column of *V*. Each such linear system can be solved using $\tilde{O}(n)$ field operations [BGY80]. As a result, U^{-1} can be computed in $\tilde{O}(n^2)$ time.

Finally, since $C = U^{-1} \cdot A \cdot U$ and all but the last column of C are fixed, we can determine C by simply computing the last column of $U^{-1} \cdot A \cdot U$, i.e., multiplying $U^{-1} \cdot A \cdot A^{n-1}u$ in $O(n^2)$ time. \Box

Lemma 3.5 can produce an FNF of a generic A in $\tilde{O}(T(n) + n^2)$ time, where T(n) is the time required to multiply a vector by the first n powers of either A or A^T . Note that A^T and A have the same characteristic and minimal polynomials, so A^T is generic iff A is generic. Thus, by Theorem 3.3, O(polylog n) samples of u, v are enough to succeed w.h.p.

The crucial property of the Frobenius normal form that we will use is that the companion matrix C is particularly easy to power and for any $k \ge 1$ we have

$$A^{k} = (UCU^{-1})^{k} = U(CU^{-1}U)^{k-1}CU^{-1} = UC^{k}U^{-1}.$$

The companion matrix C has the following key property.

Fact 3.6. For any $k \ge 1$, let $w_1, w_2, \ldots, w_n \in \mathbb{F}^{n \times 1}$ be the columns of C^k . Then we have

$$C^{k+1} = \begin{bmatrix} w_2 & w_3 & \dots & w_n & C \cdot w_n \end{bmatrix}.$$

Corollary 3.7. Let $k \ge 1$ and let u_1, \ldots, u_n be the columns of the matrix $U \cdot C^k$. Let w_n be the last column of C^k . Then

$$U \cdot C^{k+1} = \begin{bmatrix} u_2 & u_3 & \dots & u_n & U \cdot C \cdot w_n \end{bmatrix}$$

Proof. Let $w_1, w_2, \ldots, w_n \in \mathbb{F}^{n \times 1}$ be the columns of C^k . Then, by Fact 3.6:

$$U \cdot C^{k+1} = \begin{bmatrix} Uw_2 & Uw_3 & \dots & Uw_n & UCw_n \end{bmatrix} = \begin{bmatrix} u_2 & u_3 & \dots & u_n & UCw_n \end{bmatrix}.$$

By Corollary 3.7, the *n* matrices $U \cdot C$, $U \cdot C^2$, ..., $U \cdot C^n$ can be encoded concisely using only 2n - 1 column vectors, i.e., in $O(n^2)$ space. This is formally captured by the below lemma which also shows that such a representation can be computed very efficiently.

Lemma 3.8. Let the Frobenius normal form $U \cdot C \cdot U^{-1}$ of A be given. Using $\widetilde{O}(n^2)$ field operations we can compute an auxiliary matrix $(r_{i,j}) = R \in \mathbb{F}^{n \times (2n-1)}$, such that for any $k \in [n]$:

$$U \cdot C^{k} = \begin{bmatrix} r_{1,k} & r_{1,k+1} & \dots & r_{1,n+k-1} \\ r_{2,k} & r_{2,k+1} & \dots & r_{2,n+k-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,k} & r_{n,k+1} & \dots & r_{n,n+k-1} \end{bmatrix} := R_{k}.$$

Proof. Let $u_i = (u_{i,1}, \ldots, u_{i,n})$ be the *i*-th row of the matrix U. For any k > n define $u_{i,k}$ inductively:

$$u_{i,k} = -c_0 \cdot u_{i,k-n} - c_1 \cdot u_{i,k-n+1} - \dots - c_{n-1} \cdot u_{i,k-1} = -\sum_{i=0}^{n-1} c_i \cdot u_{i,k-n+i}$$

In other words, $(u_i)_{i=1}^{\infty}$ is a linearly recursive sequence of order *n*. As discussed in the proof of Lemma 3.2, for any $k \ge 0$, we have:

$$u_i \cdot C^k = (u_{i,1+k}, \dots, u_{i,n+k}).$$

In particular, for k = n - 1, we obtain:

$$u_i \cdot C^n = (u_{i,n+1}, \dots, u_{i,2n}) = (r_{i,n}, \dots, r_{i,2n-1}).$$

Since $(r_{i,1}, \ldots, r_{i,n-1}) = (u_{i,2}, \ldots, u_n)$, the *i*-th row of the matrix R can be obtained by computing the terms $n + 1, \ldots, 2n$ of the linearly recursive sequence $(u_i)_{i=1}^{\infty}$. As shown in [BM21, Theorem 3], this can be done using $\widetilde{O}(n)$ field operations. Therefore, by applying this to all *i*, computing the entire matrix R is possible using $\widetilde{O}(n^2)$ field operations.

Lemma 3.9. Let a Frobenius normal form $U \cdot C \cdot U^{-1}$ of a generic $A \in \mathbb{F}^{n \times n}$ and the associated auxiliary matrix R of Lemma 3.8 be given. Then:

- (1) For any $i, j \in [n]$, the elements $A_{i,j}, (A^2)_{i,j}, \ldots, (A^{n-1})_{i,j}$ can be computed in $\widetilde{O}(n)$ time.
- (2) For any vector $v \in \mathbb{F}^{n \times 1}$, all the vectors $v, Av, A^2v, \ldots, A^{n-1}v$ can be computed in $\widetilde{O}(n^2)$ time.

Proof. In the former item, set $v := e_j$. We have $A^k v = U \cdot C^k \cdot (U^{-1}v)$. Let us first compute $w = U^{-1}v$. In the former item, the vector w is simply the *j*-th column of U^{-1} and thus it can be read in O(n) time. In the latter item, it can be obtained in $O(n^2)$ time.

For any $i \in [n]$, $(A^k v)_i = (UC^k w)_i = (r_{i,k}, \dots, r_{i,n+k-1}) \cdot w$. Thus, we have:

$r_{i,1}$	$r_{i,2}$		$r_{i,n}$		$\left[(A^1 v)_i \right]$	
$r_{i,2}$	$r_{i,3}$		$r_{i,n+1}$		$(A^2v)_i$	
:	:	·	:	$\cdot w \equiv$		
$r_{i,n}$	$r_{i,n+1}$		$r_{i,2n-1}$		$\left\lfloor (A^n v)_i \right\rfloor$	

Note that the $n \times n$ matrix on the left-hand side above is a Hankel matrix whose 2n - 1 distinct entries come from the precomputed matrix R. As a result, the right-hand side vector can be computed using fast Hankel matrix-vector multiplication in $\widetilde{O}(n)$ time (see, e.g., [GVL13]). This gives the desired values $A_{i,j}, (A^2)_{i,j}, \ldots, (A^{n-1})_{i,j}$ in item (1). By doing this for all $i = 1, \ldots, n$, we obtain the desired vectors $v, Av, \ldots, A^{n-1}v$ in $\widetilde{O}(n^2)$ time in item (2).

4 Computing submatrices of k first powers of a generic matrix

Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix. As shown in Lemma 3.9, one can compute a certain cell (i, j) of all the powers A^1, \ldots, A^{n-1} in $\widetilde{O}(n)$ time via Hankel matrix-vector multiplication as long as a Frobenius form of A is given. In this section, we generalize this as follows. Let $1 \le h \le n$ be an integer. Let S be a subset of rows and let T be a subset of columns of A. Our goal is to compute the $S \times T$ submatrices $(A^1)_{S,T}, \ldots, (A^h)_{S,T}$. One particularly interesting case is $S = T = \{1, \ldots, n\}$, where we want to explicitly output the first h matrix powers of A. We prove:

Theorem 4.1. Let $A \in \mathbb{F}^{n \times n}$ be generic and let $U \cdot C \cdot U^{-1}$ be its Frobenius form. Let $R = (r_{i,j})$ be the associated auxiliary matrix of Lemma 3.8. Let $S, T \subseteq [n]$. Let $h = \lfloor n^{\alpha} \rfloor$, $|S| = \lfloor n^{s} \rfloor$, $|T| = \lfloor n^{t} \rfloor$ for some $\alpha, s, t \in [0, 1]$. Then, the submatrices $(A^{1})_{S,T}, \ldots, (A^{h})_{S,T}$ can be computed using $\widetilde{O}(n^{\omega(s,1-\alpha,t)+\alpha})$ field operations.

Proof. Set $\Delta = \lfloor n/h \rfloor$. Put $U^{-1} = (g_{i,j})$. For all $i \in S$ and $j \in \{0, \ldots, \Delta - 1\}$, let

$$p_{i,j}(x) = r_{i,j\cdot h+1} \cdot x + r_{i,j\cdot h+2} \cdot x^2 + \dots + r_{i,j\cdot h+(2h-1)} \cdot x^{2h-1}$$

be a polynomial. Similarly, for all $i \in T$ and $j \in \{0, \ldots, \Delta - 1\}$, let us introduce a polynomial

$$q_{j,i}(x) = g_{j \cdot h+1,i} \cdot x^{h-1} + g_{j \cdot h+2,i} \cdot x^{h-2} + \ldots + g_{j \cdot h+h,i} \cdot x^0.$$

In the above, every value $r_{,,}$ and $g_{,}$ that has not been defined is assumed to be equal to 0.

Each $p_{i,j}$ is a polynomial of degree 2h-1, and each $q_{j,i}$ is a polynomial of degree h-1. Consider the polynomial matrices $P = (p_{i,j}) \in \mathbb{F}[x]^{|S| \times \Delta}$ and $Q = (q_{j,i}) \in \mathbb{F}[x]^{\Delta \times |T|}$. The product $P \cdot Q$ can be computed in $\widetilde{O}(n^{\omega(s,1-\alpha,t)} \cdot h) = \widetilde{O}(n^{\omega(s,1-\alpha,t)+\alpha})$ time since arithmetic operations on polynomials of degree at most h can be carried out in $\widetilde{O}(h)$ time [CT65].

Now, for $i \in S$, $j \in T$, and $k \in \{h, \ldots, 2h - 1\}$ consider the coefficient $d_{i,j,k}$ of x^k in the polynomial $(P \cdot Q)_{i,j}$ of degree at most 3*h*. We have:

$$d_{i,j,k} = \sum_{t=0}^{\Delta-1} \sum_{l=1}^{h} r_{i,t\cdot h+(k-h)+l} \cdot g_{t\cdot h+l,j}.$$

Now consider the element $A_{i,j}^k$, for $k \in [h]$:

$$A_{i,j}^{k} = (UC^{k}U^{-1})_{i,j} = \sum_{z=1}^{n} (UC^{k})_{i,z} \cdot g_{z,j}$$
$$= \sum_{z=1}^{n} r_{i,z+k-1} \cdot g_{z,j} = \sum_{t=0}^{\Delta-1} \sum_{l=1}^{h} r_{i,t\cdot h+(k-1)+l} \cdot g_{t\cdot h+l,j} = d_{i,j,k+h-1}.$$

We conclude that all the required entries in the respective submatrices $S \times T$ of A^1, \ldots, A^h are encoded as coefficients of the (polynomial) entries of the matrix $P \cdot Q$.

Note that Lemma 3.8 together with Theorem 4.1 imply Theorem 1.1.

5 Maintaining an FNF under generic rank-1 updates

Let $A \in \mathbb{F}^{n \times n}$ be again a generic matrix. In this section, we consider the following problem. Let a Frobenius normal form of A be given. Suppose A is subject to a *rank-1 update*, i.e., A is replaced with $A' = A + ab^T$ for some $a, b \in \mathbb{F}^{n \times 1}$. We require that the update is also *generic*, i.e., the obtained matrix A' is also generic. We would like to recompute an FNF of the updated matrix A' faster than from scratch which would take $\tilde{O}(n^{\omega})$ time. In this section, we show:

Theorem 1.2. Let $A \in \mathbb{F}^{n \times n}$ be a generic matrix. Suppose an FNF of A and an FNF of A^T are given. Then, for any $a, b \in \mathbb{F}^{n \times 1}$ such that $A' = A + ab^T$ is generic, Frobenius normal forms of A' and $(A')^T$ can be computed explicitly in $\widetilde{O}(n^2)$ time. The algorithm succeeds with high probability.

Let us remark that the assumption that the FNFs of both A and its transpose A^T are maintained is merely for simplicity of exposition. Recall that A^T is generic if and only if A is generic.

We now describe the update procedure. Our goal is to compute, for some vectors $u, v \in \mathbb{F}^{n \times 1}$ chosen randomly, the 2n vectors $(A + ab^T)^i \cdot u$, and $v^T \cdot (A + ab^T)^i$, for $i = 0, \ldots, n - 1$. By Theorem 3.3 and Lemma 3.5, this will give an explicit FNF of $A + ab^T$ using $\tilde{O}(n^2)$ additional time with probability $\Omega(1/\log n)$. By applying the same procedure to A^T , we will compute an FNF of $(A')^T = A^T + ba^T$ as well. After trying O(polylog n) times, we will succeed with high probability.

In the following, will only focus on computing all $(A + ab^T)^i \cdot u$, since all $v^T \cdot (A + ab^T)^i$ can be computed by proceeding symmetrically with the transpose A^T .

First of all, using Lemmas 3.8 and 3.9 applied to A and its FNF, we compute the vectors $\delta_i := A^i u$, for $i = 0, \ldots, n-1$ in $\widetilde{O}(n^2)$ time. Similarly, we compute the vectors $\alpha_i := A^i a$, for $i = 0, \ldots, n-1$ within the same time bound.

Note that for any k = 0, ..., n - 1, we can expand $X_k := (A + ab^T)^k u$ as follows:

$$\begin{aligned} X_k &= (A + ab^T)^k u = A^k u + \sum_{l=1}^k \left(A^{l-1} \cdot ab^T \cdot (A + ab^T)^{k-l} \cdot u \right) \\ &= A^k u + \sum_{l=1}^k \left((A^{l-1}a) \cdot (b^T \cdot (A + ab^T)^{k-l} \cdot u) \right) \\ &= \delta_k + \sum_{l=0}^{k-1} \alpha_l \cdot (b^T \cdot X_{(k-1)-l}) \\ &= \delta_k + \sum_{l=0}^{k-1} \alpha_{(k-1)-l} \cdot (b^T \cdot X_l). \end{aligned}$$

This way we obtain a dynamic programming formula for computing the desired subsequent vectors X_0, \ldots, X_{n-1} . The right-hand side of the recurrence involves a convolution of scalars (obtained from the previous terms) with the precomputed vectors α_l . Recurrences of this kind can be evaluated efficiently using a folklore combination of FFT and a divide-and-conquer approach, as follows.

Let us initialize vectors $X'_0, \ldots, X'_{n-1} \in \mathbb{F}^{n \times n}$ whose purpose is to store partially computed vectors X_0, \ldots, X_{n-1} . Initially, put $X'_i = \delta_i$. Let us define a recursive procedure F(p,q), for $0 \le p \le q \le n-1$ such that, assuming that for all $j = p, \ldots, q$ we have:

$$X'_{j} = \delta_{j} + \sum_{l=0}^{p-1} \alpha_{(j-1)-l} \cdot (b^{T} \cdot X_{l}),$$
(2)

updates X'_p, \ldots, X'_q so that $X'_j = X_j$ for all $j = p, \ldots, q$. Observe that the prerequisite of F(0, n-1) is satisfied by the initial values of X'_i . Moreover, after F(0, n-1) completes we are done.

Let us now discuss how F(p,q) is implemented. If p = q, there is nothing to be done, as the prerequisite (2) already implies that $X'_p = X_p$. Suppose p < q and let $m = \lfloor (p+q)/2 \rfloor$. We have $p \leq m < q$. First, F(p,m) is called; note that the prerequisite (2) of that call holds. Afterwards, each vector X'_l , for $l \in \{p, \ldots, m\}$, equals the respective vector X'_l . The next step is to compute the scalars $d_l := b^T \cdot X_l$ for all such l in O((m-p)n) = O((q-p)n) time.

Subsequently, define the following polynomials $\in \mathbb{F}[x]$ of degree at most q - p:

$$Q(x) = \sum_{i=0}^{m-p} d_{p+i} \cdot x^i,$$
$$P_k(x) = \sum_{i=0}^{q-p} \alpha_{i,k} \cdot x^i,$$

where $\alpha_{i,k}$, for k = 1, ..., n, is the k-th coordinate of α_i . For each k = 1, ..., n, compute the polynomial $P_k(x)Q(x)$ using FFT [CT65] in $\widetilde{O}(q-p)$ time. Through all k, this takes $\widetilde{O}((q-p)n)$ time. Observe that the coefficient $c_{k,i}$ of x^i in $P_k(x)Q(x)$ equals

$$\sum_{l=\max(0,i-q+p)}^{\min(i,m-p)} \alpha_{i-l,k} \cdot d_{p+l} = \sum_{l=\max(p,i-q+2p)}^{\min(p+i,m)} \alpha_{i+p-l,k} \cdot d_l.$$

In particular, for j = m + 1, ..., q, $j - p - 1 \in [0, q - p - 1]$. So we get $(j - p - 1) + p = j - 1 \ge m$ and $(j - p - 1) - q + 2p \le p - 1$. Thus, the coefficient of x^{j-p-1} in $P_k(x)Q(x)$ equals

$$c_{k,j-p-1} = \sum_{l=p}^{m} \alpha_{(j-1)-l,k} \cdot d_l.$$

We conclude that the following column vectors can be retrieved from the computed polynomials:

$$\Delta_j := \begin{bmatrix} c_{1,j-p-1} & c_{2,j-p-1} & \dots & c_{n,j-p-1} \end{bmatrix}^T = \sum_{l=p}^m \alpha_{(j-1)-l} \cdot (b^T \cdot X_l).$$

After adding Δ_j to X'_j for each $j = m + 1, \ldots, q$, we have

$$X'_{j} = \delta_{j} + \sum_{l=0}^{p-1} \alpha_{(j-1)-l} \cdot (b^{T} \cdot X_{l}) + \sum_{l=p}^{m} \alpha_{(j-1)-l} \cdot (b^{T} \cdot X_{l}) = \delta_{j} + \sum_{l=0}^{m} \alpha_{(j-1)-l} \cdot (b^{T} \cdot X_{l}).$$

As a result, the prerequisite of the call F(m+1,q) is satisfied and we can call F(m+1,q) to update X'_{m+1}, \ldots, X'_q so that they store X_{m+1}, \ldots, X_q respectively. The correctness of the procedure F(p,q) follows easily by induction on q-p.

The time T(N) needed to compute F(l, r) when r - l = N clearly satisfies:

$$T(1) = O(1)$$

$$T(N) \le T(\lceil N/2 \rceil) + T(\lfloor N/2 \rfloor) + \widetilde{O}(Nn).$$

Therefore, we get $T(N) = \widetilde{O}(Nn)$. Since the root call F(0, n-1) satisfies r-l = n, all the desired vectors $X_k = (A + ab^T)^k$ for $k = 0, \ldots, n-1$ are computed in $\widetilde{O}(n^2)$ time as desired.

6 Accessing matrix powers under batch element updates

This section is devoted to proving the following lemma on computing entries of the powers of a matrix A under single-element updates to A. This result is implicit in the known works on dynamic matrix inverse [vdBNS19, San04, San05b].

Lemma 6.1. Let $A \in \mathbb{F}^{n \times n}$. Let $\Psi = \{(u_1, v_1, y_1), \dots, (u_f, v_f, y_f)\} \subseteq [n] \times [n] \times \mathbb{F}$ be such that all pairs (u_i, v_i) are distinct. Let $S = \{u_1, \dots, u_f\}$ and $T = \{v_1, \dots, v_f\}$. Suppose the matrix B is obtained from A by setting $A_{u_i, v_i} := y_i$ for all $i = 1, \dots, f$. Let $h \ge 1$ be an integer.

Given the submatrices $(A^1)_{T,S}, \ldots, (A^h)_{T,S}$, one can preprocess Ψ in $\widetilde{O}(f^{\omega} \cdot h)$ time, so that the following queries are supported. Given any $X, Y \subseteq [n]$ along with the submatrices $(A^1)_{X,S}, \ldots, (A^h)_{X,S}, (A^1)_{T,Y}, \ldots, (A^h)_{T,Y}$, and $(A^1)_{X,Y}, \ldots, (A^h)_{X,Y}$, compute the submatrices $(B^1)_{X,Y}, \ldots, (B^h)_{X,Y}$. The query time is $\widetilde{O}((\mathrm{MM}(|X|, f, |Y|) + \mathrm{MM}(f, f, \min(|X|, |Y|)) \cdot h)$.

Consider the ring $\mathbb{F}[X]/(X^{h+1})$ of polynomials in X over \mathbb{F} modulo X^{h+1} . Consider a matrix polynomial $I - X \cdot A$, where X is a variable. $I - X \cdot A$ can be also viewed as a matrix of degree ≤ 1 polynomials in X. The inverse $(I - X \cdot A)^{-1}$ exists in the ring $(\mathbb{F}[X]/(X^{h+1}))^{n \times n}$ of polynomial matrices modulo X^{h+1} and equals:

$$(I - X \cdot A)^{-1} = I + X \cdot A + X^2 \cdot A^2 + \ldots + X^h \cdot A^h.$$
 (3)

The above equation can be seen to hold by multiplying both sides (modulo X^{h+1}) by $I - X \cdot A$.

For each i = 1, ..., f, let $\delta_i := y_i - A_{u_i,v_i}$, so that B can be seen to be obtained from A by adding δ_i to a corresponding element A_{u_i,v_i} . Let us put

$$U = \begin{bmatrix} e_{u_1} & e_{u_2} & \dots & e_{u_f} \end{bmatrix} \in \mathbb{F}^{n \times f},$$

$$V = \begin{bmatrix} e_{v_1} & e_{v_2} & \dots & e_{v_f} \end{bmatrix}^T \in \mathbb{F}^{f \times n},$$

$$\Delta = \operatorname{diag}(\delta_1, \dots, \delta_f) \in \mathbb{F}^{f \times f}.$$

Then, we have $B = A + U \cdot \Delta \cdot V$. Consider the inverse of $I - X \cdot B$ in $\mathbb{F}[X]/(X^{h+1})$. We have:

$$(I - X \cdot B)^{-1} = (I - X \cdot (A + U\Delta V))^{-1} = (I - X \cdot A + U \cdot (-X\Delta) \cdot V)^{-1}$$

Put $Z := (I - X \cdot A)^{-1} = (z_{i,j})$. By the Sherman-Morrison-Woodbury formula (see, e.g., [HS81]), we have:

$$(I - X \cdot B)^{-1} = Z - Z \cdot U \cdot (I + (-X\Delta)VZU)^{-1} \cdot (-X\Delta)V \cdot Z$$
$$= Z - (ZU) \cdot (I - X\Delta(VZU))^{-1} \cdot (-X\Delta) \cdot (VZ).$$

Above, the matrix ZU (VZ) selects the subsequent columns $u_1, \ldots, u_f \in S$ (rows $v_1, \ldots, v_f \in T$, resp.) of Z and arranges them left to right (top to bottom, resp.). Similarly, $(VZU)_{i,j} = Z_{v_i,u_j}$ and thus VZU can be read from $Z_{T,S}$ in optimal $O(f^2)$ time.

The identity holds if $(I - X\Delta(VZU))$ is invertible in $\mathbb{F}[X]/(X^{h+1})$. It indeed is, as:

$$(I - X\Delta \cdot (VZU))^{-1} = \sum_{i=0}^{h} (\Delta VZU)^{i} \cdot X^{i}.$$

Moreover, since

$$\sum_{i=0}^{h} (\Delta VZU)^{i} \cdot X^{i} = \left(\prod_{j=1}^{\lceil \log h \rceil} \left(I + (\Delta XVZU)^{2^{j}}\right)\right) \mod X^{h+1},$$

the matrix $P = (I - X\Delta(VZU))^{-1}$ can be computed, given $Z_{T,S}$, using $\widetilde{O}(1)$ multiplications of $f \times f$ matrices whose entries are polynomials of degree at most h (and the polynomial arithmetic is performed modulo X^{h+1}), that is, in $\widetilde{O}(f^{\omega} \cdot h)$ time. Recall that since $Z = I + A + \ldots + A^h$, the polynomial matrix $Z_{T,S}$ is encoded using the given submatrices $(A^1)_{T,S}, \ldots, (A^h)_{T,S} \in \mathbb{F}^{f \times f}$.

Assuming the matrix P is precomputed, given $X, Y \subseteq [n]$, we can compute $(I - X \cdot B)_{X,Y}^{-1}$ in $\widetilde{O}(f^2 \cdot h)$ time using (rectangular) matrix multiplication as:

$$(I - X \cdot B)_{X,Y}^{-1} = Z_{X,Y} - (ZU)_{X,[f]} \cdot P \cdot (-X\Delta) \cdot (VZ)_{[f],Y}.$$

Here, the matrixces $(ZU)_{X,[f]}$ and $(VZ)_{[f],Y}$ can be read from the submatrices $Z_{X,S}$, $Z_{T,Y}$ respectively. The used order of multiplication depends on which of X, Y is smaller. This expression can be evaluated as $((((ZU)_{X,[f]} \cdot P) \cdot (-X\Delta)) \cdot (VZ)_{[f],Y})$ in $\widetilde{O}((\mathrm{MM}(|X|, f, f)) + \mathrm{MM}(|X|, f, |Y|)) \cdot h)$ time, or as $(ZU)_{X,[f]} \cdot (P \cdot ((-X\Delta) \cdot (VZ)_{[f],Y})) \widetilde{O}((\mathrm{MM}(f, f, |Y|)) + \mathrm{MM}(|X|, f, |Y|)) \cdot h)$ time.

Finally, recall that the submatrices $Z_{X,S}$, $Z_{T,Y}$, and $Z_{X,Y}$ are encoded by the submatrices $(A^1)_{X,S}, \ldots, (A^h)_{X,S}, (A^1)_{T,Y}, \ldots, (A^h)_{T,Y}$, and $(A^1)_{X,Y}, \ldots, (A^h)_{X,Y}$, respectively. The desired submatrices $(B^1)_{X,Y}, \ldots, (B^h)_{X,Y}$ are encoded by the obtained polynomial matrix $(I - X \cdot B)_{X,Y}^{-1}$.

7 Representing the graph with a generic matrix

Let G = (V, E) be an unweighted digraph. Let X be a set of variables $\tilde{x}_{u,v}$ indexed with pairs from $V \times V$. Let \tilde{Y} be a set of variables \tilde{y}_v indexed with V.

An $n \times n$ symbolic adjacency matrix $\tilde{A}(G) \in \mathbb{F}[\tilde{X} \cup \tilde{Y}]^{n \times n}$ of G is defined as follows:

$$\tilde{A}(G)_{u,v} = \begin{cases} \tilde{x}_{u,v} \cdot \tilde{y}_v & \text{if } u = v \text{ or } uv \in E, \\ 0 & \text{otherwise.} \end{cases}$$

This section is devoted to proving the following:

Theorem 7.1. For all $u, v \in V$, let us assign $\tilde{x}_{u,v}$ a random element $x_{u,v} \in \mathbb{F}$. Similarly, for all $v \in V$ assign \tilde{y}_v a random element $y_v \in \mathbb{F}$. Suppose all these random samples are independent.

Let $A \in \mathbb{F}^{n \times n}$ be a weighted adjacency matrix obtained from $\tilde{A}(G)$ using this assignment. Then, with probability at least $1 - n^4/|\mathbb{F}|$:

(1) A is generic.

(2) For every $u, v \in V$ and $k \in [n-1]$, $A_{u,v}^k \neq 0$ if and only if there exists a path $u \to v$ in G of length at most k.

We start by referring to the following lemma of Wiedemann [Wie86].

Lemma 7.2. [Wie86, Section V] Let $B \in \mathbb{F}^{n \times n}$ be such that all *n* leading principal minors of *B* are non-singular. Let $\tilde{y}_1, \ldots, \tilde{y}_n$ be variables. Then the discriminant of the characteristic polynomial of $B \cdot diag(\tilde{y}_1, \ldots, \tilde{y}_n)$ is a non-zero polynomial in $\tilde{y}_1, \ldots, \tilde{y}_n$ of degree no more than $2n^3$.

With Lemma 7.2 in hand, let us prove that the matrix A is generic. Let $\tilde{A}(G)_{|\tilde{X}=x} \in \mathbb{F}[\tilde{Y}]^{n \times n}$ be obtained from $\tilde{A}(G)$ by setting all $\tilde{x}_{u,v}$ to $x_{u,v}$. Let $\tilde{A}(G)_{|\tilde{Y}=1} \in \mathbb{F}[\tilde{X}]^{n \times n}$ $(\tilde{A}(G)_{|\tilde{X}=x,\tilde{Y}=1} \in \mathbb{F}^{n \times n},$ resp.) be obtained from $\tilde{A}(G)$ $(\tilde{A}(G)_{|\tilde{X}=x}, \text{ resp.})$ by setting all \tilde{y}_v to 1.

Lemma 7.3. With probability at least $1 - n^2/|\mathbb{F}|$, all leading principal minors of $\tilde{A}(G)_{|\tilde{X}=x,\tilde{Y}=1}$ are non-singular.

Proof. Identify V with [n]. Consider a $k \times k$ leading principal minor $\tilde{A}_k(G)_{|\tilde{Y}=1}$ of $\tilde{A}(G)_{|\tilde{Y}=1}$. $\det(\tilde{A}_k(G)_{|\tilde{Y}=1})$ is a polynomial of degree k in \tilde{X} containing a monomial $\prod_{i=1}^k \tilde{x}_{i,i}$, and thus is not a zero polynomial. Since the assignment $\tilde{X} = x$ is random, by the Schwartz-Zippel lemma [Sch80, Zip79], $\det(\tilde{A}_k(G)_{|\tilde{X}=x,\tilde{Y}=1}) \neq 0$ with probability at least $1-k/|\mathbb{F}| \geq 1-n/|\mathbb{F}|$. By the union bound, the probability that all the n leading principal minors are non-singular is at least $1-n^2/|\mathbb{F}|$. \Box

The following corollary proves item (1) of Theorem 7.1.

Corollary 7.4. With probability at least $1 - n^4/|\mathbb{F}|$, A is generic.

Proof. We simply apply Lemma 7.2 to the matrix $\tilde{A}(G)_{|\tilde{X}=x,\tilde{Y}=1}$ and hence obtain that the discriminant of the characteristic polynomial of $\tilde{A}(G)_{|\tilde{X}=x} = \tilde{A}(G)_{|\tilde{X}=x,\tilde{Y}=1} \cdot \operatorname{diag}(\tilde{y}_1,\ldots,\tilde{y}_n)$ is a non-zero polynomial of degree no more than $2n^3$. As a result, if one randomly assigns field elements to $\tilde{y}_1,\ldots,\tilde{y}_n$, the discriminant of the characteristic polynomial $p_A(t)$ of A is non-zero with probability at least $1 - 2n^3/|\mathbb{F}| \ge 1 - n^4/|\mathbb{F}|$ by the Schwartz-Zippel lemma. Equivalently, $p_A(t)$ has n distinct roots in an algebraically closed extension \mathbb{F} of \mathbb{F} . As the minimal and characteristic polynomials of a matrix have the same roots, we obtain that $p_A \equiv \mu_A$ when A is seen as an $n \times n$ matrix over \mathbb{F} . But it is well-known that neither the characteristic- nor the minimal polynomial of a matrix depends on the base field, and consequently $p_A \equiv \mu_A$ even if A is seen as a matrix over \mathbb{F} .

Let us now move to proving item (2) of Theorem 7.1.

Lemma 7.5. Let $u, v \in V$ and $k \in [n-1]$. Then, with probability at least $1 - 2k/|\mathbb{F}|$, $A_{u,v}^k \neq 0$ if and only if there exists a $u \to v$ path of length no more than k in G.

Proof. From the definition of matrix multiplication, one can easily prove inductively that:

$$\tilde{A}(G)_{u,v}^{k} = \sum_{\substack{(u_1,\dots,u_{k+1}) \in V^{k+1} \\ u_1 = u \\ u_k + 1 = v}} \left(\prod_{i=1}^{k} \tilde{A}(G)_{u_i,u_{i+1}}\right).$$
(4)

Observe that for a given (u_1, \ldots, u_{k+1}) in the sum above, by the definition of $\tilde{A}(G)$, the product $\prod_{i=1}^{k} \tilde{A}(G)_{u_i, u_{i+1}}$ is a non-zero monomial iff for all $i = 1, \ldots, k$, either $u_i = u_{i+1}$ or $u_i u_{i+1} \in E$.

Suppose $A(G)_{u,v}^k$ is a non-zero polynomial. Then, the sum (4) contains at least one nonzero monomial $\prod_{i=1}^k \tilde{A}(G)_{u_i,u_{i+1}}$ corresponding to a (k+1)-tuple (u_1,\ldots,u_{k+1}) with $u_1 = u$ and $u_{k+1} = v$. If one assumes that G contains self-loops, then $\prod_{i=1}^k \tilde{A}(G)_{u_i,u_{i+1}}$ certifies the existence of a $u \to v$ path consisting of k edges or self-loops in G. By eliminating the self-loops, one obtains that there exists a $u \to v$ path in G with at most k edges.

Now suppose that some shortest $u \to v$ path P in G has length $l \leq k$. We prove that $\tilde{A}(G)_{u,v}^k$ is a non-zero polynomial in that case. Let $P = e_1 \dots e_l$, where $u_i v_i = e_i \in E$. Clearly, $u_i \neq v_i$ since P is shortest. Set $u_j := v$ for all $j = l + 1, \dots, k + 1$. Note that all u_1, \dots, u_{l+1} are distinct since P cannot contain cycles. Consider the monomial $M = \left(\prod_{i=1}^l x_{u_i,u_{i+1}} \cdot y_{u_{i+1}}\right) \cdot (x_{v,v}y_v)^{k-l}$. We now argue that this monomial M appears in the sum (4) precisely once, contributed by the tuple (u_1, \dots, u_{k+1}) . For contradiction, suppose there exists some other tuple (u'_1, \dots, u'_{k+1}) contributing the same monomial M. Let j > 1 be the first index such that $u_j \neq u'_j$.

If we had $u'_j = u'_{j-1}$, then $u_j \neq u_{j-1}$ since $u_{j-1} = u'_{j-1}$. As a result, $u'_{j-1} = u_{j-1} \neq v$. Consequently, the monomial contributed by (u'_1, \ldots, u'_{k+1}) contains a variable $x_{u_{j'}, u_{j'}} \neq x_{v,v}$ that M does not contain, a contradiction.

Therefore, $u'_j \neq u'_{j-1}$. Then, the monomial contributed by (u'_1, \ldots, u'_{k+1}) contains the variable x_{u_{j-1},u'_j} . M contains only a single variable of the form $x_{u_{j-1},\cdot}$, namely x_{u_{j-1},u_j} . But $u_j \neq u'_j$, a contradiction. Therefore, M is indeed a monomial of $\tilde{A}(G)_{u,v}^k$ and thus $\tilde{A}(G)_{u,v}^k$ is non-zero.

We conclude that $\tilde{A}(G)_{u,v}^k$ is a non-zero polynomial if and only if there exists a $u \to v$ path of length at most k in G. Finally, $\tilde{A}(G)_{u,v}^k \equiv 0$ implies $A_{u,v}^k = 0$. On the other hand, if $\tilde{A}(G)_{u,v}^k \not\equiv 0$, then $\tilde{A}(G)_{u,v}^k$ has degree at most 2k, so by the Schwartz-Zippel lemma, $A_{u,v}^k \neq 0$ with probability at least $1 - 2k/|\mathbb{F}|$. We obtain that the equivalence is preserved after variable substitution with desired probability.

Item (2) of Theorem 7.1 follows by combining the above lemma for all u, v, k via the union bound – the success probability is at least $1 - n^2 \cdot 2 \cdot (1 + 2 + \ldots + (n-1))/|\mathbb{F}| \ge 1 - n^4/|\mathbb{F}|$.

7.1 Handling edge weights

Encoding distances via matrix powers crucially requires that all the edges of G have positive and equal weight, or equivalently, that G is unweighted.

For a weighted digraph G = (V, E) with *n* vertices and *m* edges with integer weights in [1, W], we can, however, construct a related *unweighted* digraph G' = (V', E') with *nW* vertices and m + n(W - 1) edges, such that:

- Each vertex $v \in V$ corresponds to W vertices v^1, \ldots, v^W in G', assembled into a directed path with W 1 edges $v^W v^{W-1}, v^{W-1} v^{W-2}, \ldots, v^2 v^1$.
- Each edge $uv \in E$ of weight c has a corresponding edge u^1v^c in G'.

Lemma 7.6. For any $u, v \in V$, $\delta_G(u, v) = \delta_{G'}(u^1, v^1)$.

Proof. If u = v, $\delta_G(u, v) = \delta_{G'}(u^1, v^1) = 0$. So let us assume $u \neq v$. Then $\delta_G(u, v)$, $\delta_{G'}(u^1, v^1) > 0$. Let us first prove $\delta_{G'}(u^1, v^1) \leq \delta_G(u, v)$. If $\delta_G(u, v) = \infty$ then this is trivial. Suppose $\delta_G(u, v) = d$. Then there exists an $u \to v$ path $P = u_1 u_2 \dots u_k$ of weight d in G, where $k \leq d + 1$.

Let
$$c_i$$
 be the weight of the edge $u_i u_{i+1}$ in G. Consider the path

$$P' = (u_1^1 u_2^{c_1} \cdot u_2^{c_1} u_2^{c_1-1} \cdot \ldots \cdot u_2^2 u_2^1) \cdot (u_2^1 u_3^{c_2} \cdot u_3^{c_2} u_2^{c_2-1} \cdot \ldots \cdot u_3^2 u_3^1) \cdot \ldots \cdot (u_{k-1}^1 u_k^{c_{k-1}} \cdot u_k^{c_{k-1}} u_k^{c_{k-1}-1} \cdot \ldots \cdot u_k^2 u_k^1).$$

By the construction of G' and the existence of P, P' exists in G' and consists of $\sum_{i=1}^{k-1} c_i = d$ edges. So indeed $\delta_{G'}(u^1, v^1) \leq \delta_G(u, v)$.

Now we prove $\delta_G(u, v) \leq \delta_{G'}(u^1, v^1)$. Again, if $\delta_{G'}(u^1, v^1) = \infty$, there is nothing to prove. Otherwise, let $\delta_{G'}(u^1, v^1) = d \geq 1$. There exists an $u^1 \to v^1$ path $Q = z_1^{p_1} \dots z_{d+1}^{p_{d+1}}$ in G', where $z_1, \dots, z_{d+1} \in V$, $z_1 = u$, $p_1 = 1$, $z_{d+1} = v$, and $p_{d+1} = 1$. Let $j_1 < \dots < j_k$ be all indices j such that $p_j = 1$. In particular, $j_1 = 1$ and $j_k = d + 1$. Since in G', a vertex $w^q \in V'$, for q > 1 has only a single outgoing edge $w^q w^{q-1}$, $j_i < l < j_{i+1}$ implies that $z_l = z_{l-1}$ and $p_l = p_{l-1} - 1$. As a result, for i > 1, $z_{j_i} = z_{j_{i-1}+1}$. We obtain that P' can be expressed as:

$$P' = (z_1^1 z_2^{p_2} z_2^{p_2 - 1} \dots z_2^1) \cdot (z_{j_2}^1 z_{j_2 + 1}^{p_{j_2 + 1}} z_{j_2 + 1}^{p_{j_2 + 1} - 1} \dots z_{j_2 + 1}^1) \cdot \dots \cdot (z_{j_{k-1}}^1 z_{j_{k-1} + 1}^{p_{j_{k-1} + 1}} z_{j_{k-1} + 1}^{p_{j_{k-1} + 1} - 1} \dots z_{j_{k-1} + 1}^1).$$

Hence we conclude that P' has $\sum_{i=1}^{k-1} p_{j_i+1}$ edges. Moreover, by the construction of G', for each $i = 1, \ldots, k-1$, there exists an edge $z_{j_i} z_{j_i+1}$ of weight p_{j_i+1} in G. As a result, there exists a path $z_{j_1} \to z_{j_k} = z_1 \to z_{d+1} = u \to v$ of weight $\sum_{i=1}^{k-1} p_{j_i+1} = |P'| = d$. This implies the desired inequality $\delta_G(u, v) \leq \delta_{G'}(u^1, v^1)$.

Finally, let us note that by the correspondence of edges in G and G', an insertion or deletion (failure) or a single edge uv of weight c in G can be reflected by a single edge insertion or deletion of the edge u^1v^c in G'. Similarly, a failure of a vertex v in G can be translated to a failure of a single vertex v^1 in G'.

8 Distance sensitivity oracles

Let G = (V, E) be a digraph. Recall that a distance sensitivity oracle (DSO) is a data structure answering queries about $\delta_{G-F}(s,t)$, where $s,t \in V$ and $F \subseteq V \cup E$. The DSO problem can also be generalized by introducing the *update* procedure that takes the set F and preprocesses the failures to support efficient queries (s,t) about $\delta_{G-F}(s,t)$ with the failures F fixed. Such a variant has been studied, e.g., in [vdBS19] and the objective is to give a tradeoff between the preprocessing, update, and query times.

In the following, we will focus, without loss of generality, on edge failures only. In directed graphs, vertex failures can be easily reduced to edge failures via a standard vertex-splitting trick, as described next. Construct a related graph G', at most twice as large as G, as follows. Each vertex v of G is split into two vertices v_{in}, v_{out} connected by a directed edge $v_{in}v_{out}$. Each edge $uv \in E$ gives rise to an edge $u_{out}v_{in}$ in G'. Every k-edge $s \to t$ path P in G corresponds to a 2k + 1-edge path $P' = s_{in} \to t_{out}$ in G' such that P goes through a vertex z in G iff P' goes through the edge $z_{in}z_{out}$ in G'. As a result, a failure of vertex z of G can be simulated using a failure of the edge $z_{in}z_{out}$ in G'. Clearly, if $F' \subseteq E(G')$ is obtained from $F \in V(G) \cup E(G)$ by replacing failing vertices with failing edges this way, then $\delta_{G-F}(s,t) = (\delta_{G'-F'}(s_{in}, t_{out}) - 1)/2$.

8.1 Single failures

Let us first consider the 1-DSO problem, i.e., we only allow queries of the form (s, t, F), where F contains a single edge f of G. [GR21, Ren22] showed the following reduction of the 1-DSO problem to the *h*-truncated 1-DSO problem where one is only interested in supporting queries computing min $(\delta_{G-f}(s,t),h)$ instead.

Theorem 8.1. [GR21, Section 3.3] Let G be an unweighted digraph. Suppose an h-truncated 1-DSO \mathcal{D}_h for G with preprocessing time P(n) and query time Q(n) is given. Then a general

Monte-Carlo randomized 1-DSO \mathcal{D} for G with O(1) query time and $\widetilde{O}(n^2)$ space can be constructed in $\widetilde{O}(P(n) + n^{2+\rho} + n^2 \cdot Q(n) + n^3/h)$ time. If \mathcal{D}_h produces correct answers w.h.p., then so does \mathcal{D} .

[GR21] showed an *h*-truncated 1-DSO with $\tilde{O}\left(n^{\omega} + n^{\omega(1,1-\alpha,1-\alpha)+2\alpha}\right)$ preprocessing time for $h = \Theta(n^{\alpha})$. This implies, by Theorem 8.1, a general 1-DSO with preprocessing time $O(n^{2.58})$ and O(1) query time if *h* is chosen appropriately. This construction time bound does not, however, match the $\tilde{O}(n^{2+\rho}) = \tilde{O}(n^{2.529})$ time bound of Zwick's APSP algorithm [Zwi02]. The *h*-truncated 1-DSO (and also the general 1-DSO) of [GR21] also generalizes to digraphs with integer weights [1, W] at the cost of an additional factor *W* in the preprocessing time. We give an improved *h*-truncated 1-DSO for unweighted digraphs, as captured by the following lemma.

Lemma 8.2. Let G be an unweighted digraph. For $h = \Theta(n^{\alpha})$, there exists an h-truncated DSO with $\widetilde{O}(n^{\omega(1,1-\alpha,1)+\alpha})$ preprocessing time, $O(n^2h)$ space and $\widetilde{O}(h)$ query time. The data structure is Monte Carlo randomized and answers queries correctly with high probability.

Proof. Fix the field \mathbb{F} to be $\mathbb{Z}/p\mathbb{Z}$ for some prime number $p = \Theta(n^{4+c})$, where $c \geq 1$ is a constant controlling the error probability. Let $A \in \mathbb{F}^{n \times n}$ be a weighted adjacency matrix of Theorem 7.1. Recall that A is generic with probability at least $1 - 1/n^c$. The preprocessing is simply to compute the matrix powers A^1, \ldots, A^h , which can be done in $\widetilde{O}(n^{\omega(1,1-\alpha,1)+\alpha})$ time by Theorem 4.1. Theorem 4.1 requires an FNF of A, which can be computed in $\widetilde{O}(n^{\omega})$ time by Lemma 3.4, and an auxiliary matrix R of Lemma 3.8, computed in $\widetilde{O}(n^2)$ time. Note that $\omega(1, 1 - \alpha, 1) + \alpha \geq \omega$.

Observe that if the graph G is subject to a failure of a single edge f = uv, the weighted adjacency matrix A – assuming the same variable substitution in $\tilde{A}(G)$ – undergoes a single element update of changing the entry $A_{u,v}$, $u \neq v$, to 0. Let B denote the matrix A after such an update. By Lemma 6.1, for any $s, t \in V$, we can compute $(B^1)_{s,t}, \ldots, (B^h)_{s,t}$ in $\tilde{O}(h)$ time. By Theorem 7.1, with high probability, if $d \leq h$ is minimal such that $(B^d)_{s,t} \neq 0$, then $\delta_{G-f}(s,t) = d$, and otherwise, if such a value d does not exists then $\delta_{G-f}(s,t) > h$ and thus $\min(\delta_{G-F}(s,t), h) = h$.

By using the above lemma for $h = \Theta(n^{1-\rho})$ and applying Theorem 8.1, we have⁵:

Theorem 1.3. Let G be an unweighted digraph. In $\tilde{O}(n^{2+\rho}) = O(n^{2.529})$ time one can construct a distance sensitivity oracle for G handling single-edge/vertex failures with O(1) query time and $\tilde{O}(n^2)$ space. The data structure is Monte Carlo randomized and the produced answers are correct with high probability.

Notably, the obtained data structure of Theorem 1.3 matches Zwick's best-known APSP bound [Zwi02] in terms of preprocessing time (up to polylog factors) and has optimal O(1) query time. As discussed in Section 7.1, the approach can be generalized to digraphs with integer weights in [1, W] at the cost of $\tilde{O}(W^{2+\rho}) = O(W^{2.529})$ multiplicative overhead in the preprocessing time.

8.2 Multiple failures

Let us now consider supporting an arbitrary number f of edge failures in the preprocess-updatequery model. We will show the following.

Theorem 1.4. Let G be an unweighted digraph. There exists a distance sensitivity oracle with $\widetilde{O}(n^{\omega})$ preprocessing and $O(n^2)$ space such that for any set F of f edge or vertex failures, the data structure can be updated in $\widetilde{O}(nf^{\omega-1})$ time to support distance queries with failures F in $\widetilde{O}(nf)$ time. The data structure is Monte Carlo randomized and the produced answers are correct w.h.p.

⁵Similarly as in [GR21, Ren22], the size of the obtained DSO is $\tilde{O}(n^2)$ even though superquadratic $\tilde{O}(n^{3-\rho})$ space is used during the construction phase.

Hitting sets. Before we continue, let us recall a standard *hitting set trick* [UY91] that proved useful in solving shortest path problems across multiple settings in the past.

Lemma 8.3. Let G be an unweighted digraph. Let $h \in [1, n]$ be an integer. Let $H \subseteq V$ be a subset of V obtained by sampling $\gamma \cdot (n/h) \log n$ vertices uniformly and independently, where $\gamma \geq 1$ is a sufficiently large constant. For any $s, t \in V$, let $G_{H,s,t}$ be a weighted digraph on $H \cup \{s,t\}$ such that for any $u, v \in V(G_{H,s,t})$, an edge uv of weight $\delta_G(u, v)$ appears in $E(G_{H,s,t})$ iff $\delta_G(u, v) \leq h$. Then, $\delta_G(s,t) = \delta_{G_{H,s,t}}(s,t)$ holds with high probability depending on the constant γ .

Lemma 8.3 reduces computing $\delta_G(s,t)$ to finding *h*-bounded distances between $\tilde{O}(n/h)$ vertices of *G*. Once the (potentially dense) auxiliary graph $G_{H,s,t}$ from Lemma 8.3 is constructed, obtaining the desired *s*, *t*-distance amounts to running Dijkstra's algorithm on $G_{H,s,t}$ in $\tilde{O}((n/h)^2)$ time. Significantly, a sampled hitting set *H* is valid for any graph on *V*, i.e., with high probability, the same $H \subseteq V$ can be used with poly(*n*) (possibly random) different graphs, as long as *H* is independent of these graphs. For example, in the dynamic setting, *H* is valid for poly(*n*) versions of the evolving graph *G* if the queries do not reveal any information about *H*. In particular, for the studied oracles computing *exact* distances, the answers are uniquely determined by the input graph and thus do not reveal the random bits behind the used hitting sets.

Preprocessing. The only preprocessing is to construct a weighted adjacency matrix A (over a sufficiently large \mathbb{F} for G as described in Theorem 7.1) and its FNF along with the auxiliary matrix R from Lemma 3.8, which costs $O(n^2)$ space. The preprocessing takes $\tilde{O}(n^{\omega})$ time by Lemma 3.4.

Update. Given a batch F of $f = \Theta(n^{\beta})$ failing edges, $\beta < 1$, we proceed as follows. Let S contain all the endpoints of the failing edges F. We have |S| = O(f). Let H be a sampled hitting set from Lemma 8.3 for $h = \lceil n/f \rceil$. As a result, $|H| = \widetilde{O}(n/h) = \widetilde{O}(f)$. Using Theorem 4.1, we compute the submatrices $(A^1)_{S \cup H}, \ldots, (A^h)_{S \cup H}$ in $\widetilde{O}(n^{\omega(\beta,\beta,\beta)+(1-\beta)}) = \widetilde{O}(n^{1+\beta\cdot(\omega-1)}) = \widetilde{O}(nf^{\omega-1})$ time.

Consider the weighted adjacency matrix B of G-F. B is obtained from A by zeroing the entries $A_{u,v}$ for all $uv \in F$. Therefore, B is obtained from A via f element updates. By Lemma 6.1, for any $x, y \in H$, the elements $(B^1)_{x,y}, \ldots, (B^h)_{x,y}$ can be computed, given the preprocessed submatrices of the powers of A, in $\widetilde{O}(f^{\omega} \cdot h) = \widetilde{O}(nf^{\omega-1})$ time. Recall that by Theorem 7.1, the submatrices $(B^1)_{H}, \ldots, (B^h)_{H}$ encode the h-bounded distance between H in G - F.

The matrices stored upon update use $\widetilde{O}(nf) = O(n^2)$ space.

Query. Suppose we want to compute $\delta_{G-F}(s,t)$ for query vertices $s,t \in V$. We construct the graph $G_{H,s,t}$ from Lemma 8.3. Observe that we have precomputed most of the edges of $G_{H,s,t}$ in the update phase. It remains to compute the (weights) of edges incident to s and t in $G_{H,s,t}$. To this end, we first compute the submatrices $(A^1)_{\{s,t\},S\cup H\cup\{s,t\}},\ldots,(A^h)_{\{s,t\},S\cup H\cup\{s,t\}}$ and the submatrices $(A^1)_{S\cup H,\{s,t\}},\ldots,(A^h)_{S\cup H,\{s,t\}}$ in $\widetilde{O}(n^{\omega(0,\beta,\beta)+(1-\beta)}) = \widetilde{O}(n^{\beta+1}) = \widetilde{O}(nf)$ time. Afterwards, we can apply Lemma 6.1 to compute the submatrices $(B^1)_{\{s,t\},H\cup\{s,t\}},\ldots,(B^h)_{\{s,t\}},\ldots,(B^h)_{\{s,t\}}$ and the submatrices $(B^1)_{H,\{s,t\}},\ldots,(B^h)_{H,\{s,t\}}$ and the submatrices $(B^1)_{H,\{s,t\}},\ldots,(B^h)_{H,\{s,t\}}$ and the submatrices is to run Dijkstra's algorithm to compute the shortest s, t path in $G_{H,s,t}$ in $\widetilde{O}(f^2)$ time.

9 Dynamic distances

In this section, we describe three different distance oracles for fully dynamic unweighted digraphs.

9.1 Tweaking the data structure of [vdBFN22]

Theorem 9.1. [vdBFN22] Let $B \in \mathbb{F}^{n \times n}$ and let $0 \leq \nu \leq \mu \leq 1$. Let $h \in [1, n]$ be an integer. Let $S, T \subseteq [n]$. There exists a data structure maintaining the $S \times T$ submatrix of the inverse of the polynomial matrix $I - X \cdot B \in \mathbb{F}[X]/(X^{h+1})$ under element updates to B and single-element changes (additions or removals) to the sets S and T as long as $|S|, |T| \leq n^{\mu}$. The initialization time is $\widetilde{O}(h \cdot n^{\omega})$ and the worst-case update time is $\widetilde{O}((n^{\omega(1,\mu,1)-\mu} + n^{\omega(1,\nu,\mu)-\nu} + n^{\mu+\nu} + |S| \cdot |T|) \cdot h)$.

The data structure of Theorem 9.1 can be used to maintain an unweighted digraph G under single-edge insertions and deletions and answer s, t-distance queries in G as follows (see also [vdBFN21, Section C]). For a parameter $\lfloor n^{\alpha} \rfloor = h \in [1, n]$, sample a random hitting set $H \subseteq V$ of size $\tilde{\Theta}(n/h)$ as in Lemma 8.3. The data structure of Theorem 9.1 is set up for the (unweighted) adjacency matrix A^* of G and S = T = H, and the field \mathbb{F} is chosen to be $\mathbb{Z}/p\mathbb{Z}$ for a sufficiently large random prime $p \in n^{\Theta(1)}$. As discussed in the proof of Lemma 6.1, the maintained $H \times H$ submatrix of $(I - XA^*)^{-1}$, encodes the submatrices $((A^*)^1)_H, \ldots, ((A^*)^h)_H$. Those, in turn, encode, with high probability, the h-bounded distances between the vertices H in G.

To compute $\delta_G(s,t)$ for query vertices $s, t \in V$, we first temporarily add s, t to the sets S and T, at the cost of O(1) updates issued to the data structure. Afterwards, the maintained submatrix can be used to construct the graph $G_{H,s,t}$ of Lemma 8.3, and consequently $\delta_G(s,t)$ can be computed in $\widetilde{O}((n/h)^2)$ additional time by running Dijkstra's algorithm on $G_{H,s,t}$. After $\delta_G(s,t)$ is computed, we remove the temporarily added vertices $\{s,t\} \setminus H$ from S and T.

Both updates and queries are processed in $\widetilde{O}(n^{\omega(1,\mu,1)-\mu+\alpha} + n^{\omega(1,\nu,\mu)-\nu+\alpha} + n^{\mu+\nu+\alpha} + n^{2-\alpha})$ worst-case time. By setting $\mu = 0.862$, $\nu = 0.543$, and $h = n^{0.297}$, [vdBFN22] obtain $\widetilde{O}(n^{1.703})$ update and query bound.⁶

Using Theorems 7.1 and 4.1, we can obtain an improved bound by slightly altering how the data structure behind Theorem 9.1 operates when initialized with the weighted adjacency matrix A from Theorem 7.1 instead of the unweighted adjacency matrix A^* . Specifically, the dynamic matrix inverse data structure of Theorem 9.1 operates, at the topmost level, in phases of $\Theta(n^{\mu})$ element updates. At the end of each phase, the inverse $(I - XA)^{-1}$ is explicitly recomputed from the inverse at the beginning of the phase and the $\Theta(n^{\mu})$ most recent updates using fast rectangular matrix multiplication in $\widetilde{O}(n^{\omega(1,\mu,1)} \cdot h)$ time. In a standard way, this cost can be distributed over the $\Theta(n^{\mu})$ updates and hence the $\widetilde{O}(n^{\omega(1,\mu,1)-\mu} \cdot h)$ term in the update bound. However, we can as well recompute $(I - XA)^{-1}$ (mod X^{h+1}) from scratch using Theorem 4.1 in $\widetilde{O}(n^{\omega(1,1-\alpha,1)+\alpha})$ time as

$$(I - XA)^{-1} \mod X^{h+1} = I + X \cdot A + \ldots + X^h \cdot A^h$$

(see Section 6). Since this recomputation happens every $\Theta(n^{\mu})$ updates, we obtain a slightly different $\widetilde{O}(n^{\omega(1,1-\alpha,1)-\mu+\alpha} + n^{\omega(1,\nu,\mu)-\nu+\alpha} + n^{\mu+\nu+\alpha} + n^{2-\alpha})$ update/query bound for $h = n^{\alpha}$, as long as $1 - \alpha \leq \mu$. By using the online term balancing tool [Bra], we find that for $\mu = 0.793$, $\nu = 0.552$, and $\alpha = 0.328$, the bound is $O(n^{1.673})$.

Theorem 1.5. Let G be an unweighted digraph. There exists a Monte Carlo randomized data structure maintaining G under single-edge insertions and deletions and supporting s,t-distance queries with $O(n^{1.673})$ worst-case update and query time. The answers produced are correct w.h.p.

9.2 Another data structure for single-edge updates

If $\omega = 2$, both our data structure of Theorem 1.5 and that of [vdBFN22] yield an $\tilde{O}(n^{1+5/8})$ update/query bound if properly optimized. In this section, we show a different dynamic distance oracle summarized as follows.

⁶One can use the online term balancer [Bra] to reproduce this bound for the given parameters.

Theorem 1.6. Let G be an unweighted digraph. There exists a Monte Carlo randomized data structure maintaining G under single-edge insertions and deletions and supporting s,t-distance queries with $\widetilde{O}\left(n^{\frac{\omega+1}{2}}\right)$ worst-case update and query time. The answers produced are correct w.h.p.

The bound $\tilde{O}\left(n^{\frac{\omega+1}{2}}\right) = O(n^{1.687})$ of Theorem 1.6 is slightly worse than the $O(n^{1.673})$ bound obtained in Theorem 1.5, but leads to a more natural $O(n^{1.5})$ bound under the optimistic assumption $\omega = 2$. Moreover, it breaks through the theoretical $O(n^{1+5/8})$ limit of the other discussed approaches already if $\omega < 2.25$.

Update. The algorithm operates in phases of $\lceil n^{1-\alpha} \rceil$ edge updates, for $\alpha \in [0, 1]$ to be set later. At any point of time, we denote by A the weighted adjacency matrix (see Theorem 7.1) of the graph G from the beginning of the current phase, and by B a weighted adjacency matrix of the current graph G. The matrix B equals A immediately after the phase starts and evolves by single-element updates corresponding to the edge updates issued to G. In particular, if an edge uv is inserted into G, a fresh random field element $x_{u,v}$ is sampled to guarantee that B is always obtained from $\tilde{A}(G)$ via random variable substitution (see Theorem 7.1).

When a phase starts, we compute in $O(n^{\omega})$ time the weighted adjacency matrix A of the graph G along with an FNF of A (Lemma 3.4) and the auxiliary matrix R of Lemma 3.8. This costly computation happens once per phase and thus takes $\widetilde{O}(n^{\omega-1+\alpha})$ amortized time per update. Moreover, for $h = \lceil n^{\alpha} \rceil$ we also sample a hitting set $H \subseteq V$ of size $\Theta(n/h \log n) = \widetilde{O}(n^{1-\alpha})$.

When a phase proceeds, let us denote by $S \subseteq V$ the set of endpoints of the edges inserted or deleted in the current phase. At the beginning of a phase, $S = \emptyset$ and we always have $|S| = O(n^{1-\alpha})$. Throughout, we make sure that all the submatrices $(A^1)_{H\cup S}, \ldots, (A^h)_{H\cup S}$ are stored explicitly. To this end, when a phase starts, we compute the submatrices $(A^1)_H, \ldots, (A^h)_H$. This takes $\widetilde{O}(n^{\omega(1-\alpha,1-\alpha,1-\alpha)+\alpha}) = \widetilde{O}(n^{(1-\alpha)\omega+\alpha}) \subseteq \widetilde{O}(n^{\omega})$ time by Theorem 4.1. Amortized over the $\Theta(n^{1-\alpha})$ updates in a phase, this costs $\widetilde{O}(n^{\omega-1+\alpha})$ time. Upon an update of an edge uv, u and v are inserted into S, so we only need to compute the submatrices $((A^j)_{\{u,v\},H\cup S\cup \{u,v\}})_{j=1}^h$ and $((A^j)_{H\cup S,\{u,v\}})_{j=1}^h$ to satisfy the invariant. This costs $\widetilde{O}(n^{\omega(0,1-\alpha,1-\alpha)+\alpha}) = \widetilde{O}(n^{2-\alpha})$ time by Theorem 4.1.

We also maintain the submatrices $(B^1)_H, \ldots, (B^h)_H$. They are initialized trivially to the corresponding computed submatrices $(A^1)_H, \ldots, (A^h)_H$ when a phase starts. By Lemma 6.1, they can be updated subject to an element change (u, v) (corresponding to an insertion or deletion of the edge uv in G) in B in $\widetilde{O}(|H|^2 \cdot h) = \widetilde{O}(n^{2-\alpha})$ time if the (current) submatrices $(B^1)_{H\cup\{u,v\}}, \ldots, (B^h)_{H\cup\{u,v\}})_{j=1}^h$ are provided. To provide those, we only need to construct the submatrices $((B^j)_{\{u,v\},H\cup\{u,v\}})_{j=1}^h$, as the other entries are maintained explicitly. Again, by Lemma 6.1, those can be obtained from the submatrices $((A^j)_{H\cup S\cup\{u,v\}})_{j=1}^h$ (that are off by at most |S| element updates to A) in $\widetilde{O}((\mathrm{MM}(2,|S|,|H|) + \mathrm{MM}(|S|,|S|,2)) \cdot h) = \widetilde{O}(n^{2-\alpha} \cdot n^{\alpha}) = \widetilde{O}(n^{2-\alpha})$ time.

Query. Finally, to answer a distance query, we construct a graph $G_{H,s,t}$ of Lemma 8.3 and run Dijkstra's algorithm on it in $\tilde{O}(n^{2-2\alpha})$ time. As the *h*-bounded distances between the vertices Hare all encoded in the maintained submatrices $(B^1)_H, \ldots, (B^h)_H$, we only need to compute *h*bounded distances between $\{s,t\}$ and $H \cup \{s,t\}$. These, again, can be devised from the submatrices $((B^j)_{\{s,t\},H\cup\{u,v\}})_{j=1}^h$ and $((B^j)_{H,\{s,t\}})_{j=1}^h$. To construct those, we proceed identically as if an update of the edge *st* was issued: we can temporarily add $\{s,t\}$ to *S*, recompute the missing submatrices of the powers of *A* and *B*, and revert this process at the end. This way, constructing the O(|H|) missing edges of $G_{H,s,t}$ takes $\tilde{O}(n^{2-\alpha})$ time. The amortized update time of the data structure is $\tilde{O}(n^{2-\alpha} + n^{\omega-1+\alpha})$, which is optimized for $\alpha = \frac{3-\omega}{2}$. Observe that the heavy $\tilde{O}(n^{\omega})$ -time computation, the only source of amortization here, happens only when a phase starts, once per $\lceil n^{1-\alpha} \rceil$ updates. As a result, the amortized bound can be converted into a worst-case bound using a standard technique, see, e.g., [ACK17, vdBNS19].

9.3 Vertex updates

Finally, we show that the dynamic Frobenius form algorithm of Section 5 leads to the first distance oracle supporting distance queries in $\widetilde{O}(n)$ time and vertex updates (i.e., changing all the edges incident to a single vertex) significantly faster than $\widetilde{O}(n^{\omega})$ in the worst-case. We note that a static distance oracle supporting queries in $\widetilde{O}(n)$ time can be constructed in $\widetilde{O}(n^{\omega})$ time [YZ05].

Theorem 1.7. Let G be an unweighted digraph. There exists a Monte Carlo randomized data structure maintaining G under fully dynamic vertex updates in $\tilde{O}(n^2)$ worst-case time per update and supporting arbitrary pair distance queries in $\tilde{O}(n)$ time. The answers are correct w.h.p.

Proof. The data structure is very simple. We maintain a weighted adjacency matrix A of G, as given by Theorem 7.1. We also maintain a Frobenius form of A and A^T . Since updating all the incoming edges or all the outgoing edges of a vertex $v \in V(G)$ can be encoded using a rank-1 update of A, a vertex update translates to at most 2 rank-1 updates of A. Hence, by Theorem 1.2, the Frobenius forms of A and A^T can be updated subject to a vertex update on G in $\tilde{O}(n^2)$ time. After each update, we also recompute the auxiliary matrix R of Lemma 3.8 in $\tilde{O}(n^2)$ time. Given an FNF and the auxiliary matrix, for any $s, t \in V$, we can compute the entries $(A^1)_{s,t}, \ldots, (A^{n-1})_{s,t}$ in $\tilde{O}(n)$ time. By Theorem 7.1, w.h.p., $\delta_G(s, t)$ equals the minimal $d \geq 0$ such that $(A^d)_{s,t} \neq 0$.

Acknowledgment

We would like to thank Maciej Gałązka for important clarifications regarding linear algebra, and anonymous FOCS reviewers for valuable comments.

References

- [ACK17] Ittai Abraham, Shiri Chechik, and Sebastian Krinninger. Fully dynamic all-pairs shortest paths with worst-case update-time revisited. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, pages 440–452. SIAM, 2017.
- [AIMN91] Giorgio Ausiello, Giuseppe F. Italiano, Alberto Marchetti-Spaccamela, and Umberto Nanni. Incremental algorithms for minimal length paths. J. Algorithms, 12(4):615–638, 1991.
- [AvdB23] Anastasiia Alokhina and Jan van den Brand. Fully dynamic shortest path reporting against an adaptive adversary. *CoRR*, abs/2304.07403, 2023.
- [BCC⁺22] Davide Bilò, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Deterministic sensitivity oracles for diameter, eccentricities and all pairs distances. In 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, volume 229 of LIPIcs, pages 22:1–22:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

- [Ber09] Aaron Bernstein. Fully dynamic (2 + epsilon) approximate all-pairs shortest paths with fast query and close to linear update time. In 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, pages 693–702. IEEE Computer Society, 2009.
- [Ber16] Aaron Bernstein. Maintaining shortest paths under deletions in weighted directed graphs. SIAM J. Comput., 45(2):548–574, 2016.
- [BGL03] Richard P. Brent, Shuhong Gao, and Alan G. B. Lauder. Random krylov spaces over finite fields. *SIAM J. Discret. Math.*, 16(2):276–287, 2003.
- [BGY80] Richard P. Brent, Fred G. Gustavson, and David Y. Y. Yun. Fast solution of toeplitz systems of equations and computation of padé approximants. J. Algorithms, 1(3):259– 295, 1980.
- [BHG⁺21] Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In *Proceedings of the 2021 ACM-SIAM Symposium* on Discrete Algorithms, SODA 2021, pages 1836–1855. SIAM, 2021.
- [BHS07] Surender Baswana, Ramesh Hariharan, and Sandeep Sen. Improved decremental algorithms for maintaining transitive closure and all-pairs shortest paths. J. Algorithms, 62(2):74–92, 2007.
- [BJN94] Phani Bhushan Bhattacharya, Surender Kumar Jain, and SR Nagpaul. *Basic abstract algebra*. Cambridge University Press, 1994.
- [BK09] Aaron Bernstein and David R. Karger. A nearly optimal oracle for avoiding failed vertices and edges. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pages 101–110. ACM, 2009.
- [BM21] Alin Bostan and Ryuhei Mori. A simple and fast algorithm for computing the N-th term of a linearly recurrent sequence. In 4th Symposium on Simplicity in Algorithms, SOSA 2021, pages 118–132. SIAM, 2021.
- [Bra] Jan van den Brand. Complexity term balancer. www.ocf.berkeley.edu/~vdbrand/complexity/. Tool to balance complexity terms depending on fast matrix multiplication.
- [CC20] Shiri Chechik and Sarel Cohen. Distance sensitivity oracles with subcubic preprocessing time and fast query time. In Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, pages 1375–1388. ACM, 2020.
- [Che15] Shiri Chechik. Approximate distance oracles with improved bounds. In Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, pages 1–10. ACM, 2015.
- [CLPR12] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012.
- [CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

- [CWX21] Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Algorithms, reductions and equivalences for small weight variants of all-pairs shortest paths. In 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, volume 198 of LIPIcs, pages 47:1–47:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [CZ] Shiri Chechik and Tianyi Zhang. Faster Deterministic Worst-Case Fully Dynamic All-Pairs Shortest Paths via Decremental Hop-Restricted Shortest Paths, pages 87–99.
- [CZ22] Shiri Chechik and Tianyi Zhang. Nearly 2-approximate distance oracles in subquadratic time. In Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, pages 551–580. SIAM, 2022.
- [DI04] Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. J. ACM, 51(6):968–992, 2004.
- [DI05] Camil Demetrescu and Giuseppe F. Italiano. Trade-offs for fully dynamic transitive closure on dags: breaking through the $o(n^2 \text{ barrier. } J. ACM, 52(2):147-156, 2005.$
- [DP09] Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, pages 506–515. SIAM, 2009.
- [DR22] Ran Duan and Hanlin Ren. Maintaining exact distances under multiple edge failures. In STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, pages 1093–1101. ACM, 2022.
- [DZ17] Ran Duan and Tianyi Zhang. Improved distance sensitivity oracles via tree partitioning. In Algorithms and Data Structures - 15th International Symposium, WADS 2017, volume 10389 of Lecture Notes in Computer Science, pages 349–360. Springer, 2017.
- [Ebe00] Wayne Eberly. Asymptotically efficient algorithms for the frobenius form. Technical report, Department of Computer Science, University of Calgary, 2000.
- [EFGW21] Jacob Evald, Viktor Fredslund-Hansen, Maximilian Probst Gutenberg, and Christian Wulff-Nilsen. Decremental APSP in unweighted digraphs versus an adaptive adversary. In 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, volume 198 of LIPIcs, pages 64:1–64:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [FS11] Gudmund Skovbjerg Frandsen and Piotr Sankowski. Dynamic normal forms and dynamic characteristic polynomial. *Theor. Comput. Sci.*, 412(16):1470–1483, 2011.
- [Gie95] Mark Giesbrecht. Nearly optimal algorithms for canonical matrix forms. SIAM J. Comput., 24(5):948–969, 1995.
- [GR21] Yong Gu and Hanlin Ren. Constructing a distance sensitivity oracle in O(n².5794 M) time. In 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, volume 198 of LIPIcs, pages 76:1–76:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

- [GU18] Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1029–1046. SIAM, 2018.
- [GVL13] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.
- [GW20a] Fabrizio Grandoni and Virginia Vassilevska Williams. Faster replacement paths and distance sensitivity oracles. *ACM Trans. Algorithms*, 16(1):15:1–15:25, 2020.
- [GW20b] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Fully-dynamic all-pairs shortest paths: Improved worst-case time and space bounds. In *Proceedings of the* 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, pages 2562–2574. SIAM, 2020.
- [Har09] Nicholas J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39(2):679–702, 2009.
- [HP98] Xiaohan Huang and Victor Y. Pan. Fast rectangular matrix multiplication and applications. J. Complex., 14(2):257–299, 1998.
- [HS81] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. SIAM Review, 23(1):53–60, 1981.
- [JV05] Claude-Pierre Jeannerod and Gilles Villard. Essentially optimal computation of the inverse of generic polynomial matrices. J. Complex., 21(1):72–86, 2005.
- [Kar21] Adam Karczmarz. Fully dynamic algorithms for minimum weight cycle and related problems. In 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, volume 198 of LIPIcs, pages 83:1–83:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [Kel85] Walter Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theor. Comput. Sci.*, 36:309–317, 1985.
- [KŁ19] Adam Karczmarz and Jakub Łącki. Reliable hubs for partially-dynamic all-pairs shortest paths in directed graphs. In 27th Annual European Symposium on Algorithms, ESA 2019, volume 144 of LIPIcs, pages 65:1–65:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- [KS02] Valerie King and Garry Sagert. A fully dynamic algorithm for maintaining the transitive closure. J. Comput. Syst. Sci., 65(1):150–167, 2002.
- [KS23] Adam Karczmarz and Piotr Sankowski. Fully dynamic shortest paths and reachability in sparse digraphs. In 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, volume 261 of LIPIcs, pages 84:1–84:20. Schloss Dagstuhl
 - Leibniz-Zentrum für Informatik, 2023.
- [LPW20] Andrea Lincoln, Adam Polak, and Virginia Vassilevska Williams. Monochromatic triangles, intermediate matrix products, and convolutions. In 11th Innovations in Theoretical Computer Science Conference, ITCS 2020, volume 151 of LIPIcs, pages 53:1–53:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

- [MS04] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In 45th Symposium on Foundations of Computer Science (FOCS 2004), pages 248–255. IEEE Computer Society, 2004.
- [NP95] Peter M. Neumann and Cheryl E. Praeger. Cyclic matrices over finite fields. *Journal* of the London Mathematical Society, 52(2):263–284, 1995.
- [PR14] Mihai Patrascu and Liam Roditty. Distance oracles beyond the thorup-zwick bound. SIAM J. Comput., 43(1):300–311, 2014.
- [Ren22] Hanlin Ren. Improved distance sensitivity oracles with subcubic preprocessing time. J. Comput. Syst. Sci., 123:159–170, 2022.
- [RZ11] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011.
- [San04] Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In 45th Symposium on Foundations of Computer Science, FOCS 2004, pages 509–517. IEEE Computer Society, 2004.
- [San05a] Piotr Sankowski. Shortest paths in matrix multiplication time. In Algorithms ESA 2005, 13th Annual European Symposium, volume 3669 of Lecture Notes in Computer Science, pages 770–778. Springer, 2005.
- [San05b] Piotr Sankowski. Subquadratic algorithm for dynamic shortest distances. In Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, volume 3595 of Lecture Notes in Computer Science, pages 461–470. Springer, 2005.
- [San07] Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, pages 118–126. SIAM, 2007.
- [Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. J. ACM, 27(4):701–717, 1980.
- [Som14] Christian Sommer. Shortest-path queries in static networks. *ACM Comput. Surv.*, 46(4):45:1–45:31, 2014.
- [Sto01] Arne Storjohann. Deterministic computation of the frobenius form. In 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, pages 368–377. IEEE Computer Society, 2001.
- [Sto15] Arne Storjohann. On the complexity of inverting integer and polynomial matrices. Comput. Complex., 24(4):777–821, 2015.
- [SW19] Piotr Sankowski and Karol Wegrzycki. Improved distance queries and cycle counting by frobenius normal form. *Theory Comput. Syst.*, 63(5):1049–1067, 2019.
- [Tho04] Mikkel Thorup. Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, volume 3111 of Lecture Notes in Computer Science, pages 384–396. Springer, 2004.

- [Tho05] Mikkel Thorup. Worst-case update times for fully-dynamic all-pairs shortest paths. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing 2005*, pages 112–119. ACM, 2005.
- [TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. J. ACM, 52(1):1–24, 2005.
- [UY91] Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991.
- [vdB21] Jan van den Brand. Unifying matrix data structures: Simplifying and speeding up iterative algorithms. In 4th Symposium on Simplicity in Algorithms, SOSA 2021, pages 1–13. SIAM, 2021.
- [vdBFN21] Jan van den Brand, Sebastian Forster, and Yasamin Nazari. Fast deterministic fully dynamic distance approximation. *CoRR*, abs/2111.03361, 2021.
- [vdBFN22] Jan van den Brand, Sebastian Forster, and Yasamin Nazari. Fast deterministic fully dynamic distance approximation. In 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, pages 1011–1022. IEEE, 2022.
- [vdBN19] Jan van den Brand and Danupon Nanongkai. Dynamic approximate shortest paths and beyond: Subquadratic and worst-case update time. In 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, pages 436–455. IEEE Computer Society, 2019.
- [vdBNS19] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, pages 456–480. IEEE Computer Society, 2019.
- [vdBS19] Jan van den Brand and Thatchaphol Saranurak. Sensitive distance and reachability oracles for large batch updates. In 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, pages 424–435. IEEE Computer Society, 2019.
- [Vil00] Gilles Villard. Computing the frobenius normal form of a sparse matrix. In Computer Algebra in Scientific Computing, pages 395–407, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [Wie86] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans.* Inf. Theory, 32(1):54–62, 1986.
- [Wul12] Christian Wulff-Nilsen. Approximate distance oracles with improved preprocessing time. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 202–208. SIAM, 2012.
- [WY13] Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Trans. Algorithms*, 9(2):14:1–14:13, 2013.
- [YZ05] Raphael Yuster and Uri Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), pages 389–396. IEEE Computer Society, 2005.

- [Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, volume 72 of Lecture Notes in Computer Science, pages 216– 226. Springer, 1979.
- [ZLS15] Wei Zhou, George Labahn, and Arne Storjohann. A deterministic algorithm for inverting a polynomial matrix. J. Complex., 31(2):162–173, 2015.
- [Zwi02] Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. J. ACM, 49(3):289–317, 2002.