Dynamic Meta-Theorems for Distance and Matching

Samir Datta
Chennai Mathematical Institute, India

Chetan Gupta
Aalto University, Finland

Rahul Jain
Fernuniversität in Hagen, Germany

Anish Mukherjee
University of Warsaw, Poland
IDEAS NCBR, Warsaw, Poland

Vimal Raj Sharma
Indian Institute of Technology, Kanpur, India

Raghunath Tewari
Indian Institute of Technology, Kanpur, India

Abstract

Reachability, distance, and matching are some of the most fundamental graph problems that have been of particular interest in dynamic complexity theory in recent years [8, 13, 11]. Reachability can be maintained with first-order update formulas, or equivalently in DynFO in general graphs with n nodes [8], even under $O(\frac{\log n}{\log \log n})$ changes per step [13]. In the context of how large the number of changes can be handled, it has recently been shown [11] that under a polylogarithmic number of changes, reachability is in $\mathsf{DynFO}[\oplus](\leq, +, \times)$ in planar, bounded treewidth, and related graph classes – in fact in any graph where small *non-zero circulation weights* can be computed in NC.

We continue this line of investigation and extend the meta-theorem for reachability to distance and bipartite maximum matching with the same bounds. These are amongst the most general classes of graphs known where we can maintain these problems deterministically without using a majority quantifier and even maintain witnesses. For the bipartite matching result, modifying the approach from [15], we convert the static non-zero circulation weights to dynamic matching-isolating weights.

While reachability is in $\mathsf{DynFO}(\leq, +, \times)$ under $O(\frac{\log n}{\log \log n})$ changes, no such bound is known for either distance or matching in any non-trivial class of graphs under non-constant changes. We show that, in the same classes of graphs as before, bipartite maximum matching is in $\mathsf{DynFO}(\leq, +, \times)$ under $O(\frac{\log n}{\log \log n})$ changes per step. En route to showing this we prove that the rank of a matrix can be maintained in $\mathsf{DynFO}(\leq, +, \times)$, also under $O(\frac{\log n}{\log \log n})$ entry changes, improving upon the previous $O(1)$ bound [8]. This implies a similar extension for the non-uniform DynFO bound for maximum matching in general graphs and an alternate algorithm for maintaining reachability under $O(\frac{\log n}{\log \log n})$ changes [13].

2012 ACM Subject Classification Theory of computation → Complexity theory and logic; Theory of computation → Finite Model Theory

Keywords and phrases Dynamic Complexity, Distance, Matching, Derandomization, Isolation, Matrix Rank

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.118

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2109.01875

Funding *Samir Datta*: Partially funded by a grant from Infosys foundation and SERB-MATRICS grant MTR/2017/000480.
Chetan Gupta: Supported by Academy of Finland, Grant 321901.
Anish Mukherjee: Partally supported by the ERC CoG grant TUgbOAT no 772346.
Vimal Raj Sharma: Ministry of Electronics and IT, India, VVS PhD program.
Raghunath Tewari: Young Faculty Research Fellowship, Ministry of Electronics and IT, India.

Acknowledgements SD and AM would like to thank Nils Vortmeier and Thomas Zeume for their comments on Section 7.

1 Introduction

In traditional complexity theory it is assumed that the given input remains fixed throughout the computation. However in real-life scenarios, many situations involve an evolving input where parts of the data change frequently. Recomputing everything from scratch for these large datasets after every change is not an efficient option. Therefore, the goal is to dynamically maintain some auxiliary data structure to help us recompute the results quickly.

The dynamic complexity framework of Patnaik and Immerman [28] is one such approach that has its roots in descriptive complexity [23] and is closely related to the setting of Dong, Su, and Topor [14]. Here we would like to make the updates and queries by first-order logic formulas. For example, by maintaining some auxiliary relations, the reachability relation can be updated after every single edge modification in FO [8] i.e., the reachability query is contained in the dynamic complexity class DynFO [28]. The motivation to use first-order logic as the update method has connections to other areas. From the circuit complexity perspective, this implies that such queries are highly parallelizable, i.e., can be updated by polynomial-size circuits in constant-time due to the correspondence between FO and uniform AC^0 circuits [2]. From the perspective of database theory, such a program can be translated into equivalent SQL queries.

The area has seen renewed interest in proving further upper bounds results, partly after the resolution of the long-standing conjecture [28] that reachability is in DynFO under single edge modifications [8]. A natural direction to extend this result is to see which other fundamental graph problems also admit such efficient dynamic programs. The closely related problems of maintaining distance and matching are two such examples, though a DynFO bound for these problems in general graphs has been elusive so far. The best known bound for distance is DynTC^0 [20] and non-uniform $\mathsf{DynAC}^0[\oplus]$ [5]. Here, the updates are computed in FO formulas with majority quantifiers (uniform TC^0 circuits) and non-uniform FO formulas with parity quantifiers ($\mathsf{AC}^0[\oplus]$ circuits), respectively. For matching, we have a non-uniform DynFO bound for maintaining the size of the maximum matching [8]. The only non-trivial class of graphs where both these problems are in DynFO is bounded treewidth graphs [12].

At the same time progress has been made to understand how large a modification to the input can be handled by similar dynamic programs. It is of particular interest since, in applications, changes to a graph often come as a bulk set of edges. It was shown that reachability can be maintained in $\mathsf{DynFO}(\leq, +, \times)$ under changes of size $O(\log n/ \log\log n)$ [13] in graphs with n nodes. Here, the class $\mathsf{DynFO}(\leq, +, \times)$ extends DynFO by access to built-in arithmetic, which is more natural for bulk changes. To handle larger changes, it is known that even for reachability, changes of size larger than polylogarithmic cannot be handled in DynFO [11]. And for changes of polylogarithmic size, the previous techniques seem to require extending DynFO by majority quantifiers [13]. Under bulk changes of polylog(n) size, we can even maintain distance and the size of a maximum matching in the uniform and the non-uniform version of $\mathsf{DynFO[MAJ]}(\leq, +, \times)$, respectively [13, 26].

Making further progress in this direction, recently it has been shown in [11] that reachability is in $\mathsf{DynFO}[\oplus](\leq,+,\times)$ (i.e., update formulas may use parity quantifiers) under polylog(n) changes in the class of graphs where polynomially bounded *non-zero circulation weights* can be computed statically in the parallel complexity class NC. A weight function for the edges of a graph has non-zero circulation if the (alternate) sum of the weights of the edges of every directed cycle is non-zero (see Section 2 for more details). Planar [29], bounded genus [10], bounded treewidth [11], single crossing minor-free [4] are some of the well-studied graph classes for which non-zero circulation weights can be computed in NC.

In this work, we first extend this result to prove similar meta-theorems for maintaining distance (including a shortest path witness) and the *search* version of minimum weight bipartite maximum matching (MinWtBMMSearch) in the same classes of graphs.

▶ **Theorem 1.** Distance *and* MinWtBMMSearch *are in* $\mathsf{DynFO}[\oplus](\leq,+,\times)$ *under polylog*(n) *edge changes on classes of graphs where non-zero circulation weights can be computed in* NC*.*

Note that these are the only classes of graphs known where we can maintain both these problems *deterministically* without using a majority quantifier and even maintain a witness to the solution (in other words, maintain a solution to the search problem).

While reachability can be maintained in $\mathsf{DynFO}(\leq,+,\times)$ under bulk changes in general graphs, no such bound is known for either distance or matching in any non-trivial class of graphs under a non-constant number of changes. Since maintaining the size of maximum matching reduces to maintaining the rank of a matrix via bounded expansion first-order truth-table (bfo-tt) reduction [8], the following gives a non-uniform $\mathsf{DynFO}(\leq,+,\times)$ bound for maintaining the size of maximum matching in *general* graphs to $O(\frac{\log n}{\log\log n})$ changes.

▶ **Theorem 2.** *Rank of a matrix from* $\mathbb{Z}_p^{n\times n}$ *is in* $\mathsf{DynFO}(\leq,+,\times)$ *under* $O(\frac{\log n}{\log\log n})$ *entry changes.*

Earlier it was known that the rank of a matrix with small integer entries can be maintained in DynFO under changes that affect a single entry [8]. As reachability reduces to matrix rank via bfo-reduction [8], Theorem 2 also gives an alternative algorithm for maintaining reachability in $\mathsf{DynFO}(\leq,+,\times)$ under $O(\frac{\log n}{\log\log n})$ changes. This is interesting in its own right as it generalizes the *rank-method* for maintaining reachability [8] even under bulk changes without going via the Sherman-Morrison-Woodbury identity [13].

Finally, building on Theorem 2, we show another meta theorem for maintaining the size of a maximum matching in bipartite graphs (BMMSize) in $\mathsf{DynFO}(\leq,+,\times)$ under slightly sublogarithmic bulk changes, in the same class of graphs as in Theorem 1. Previously, no $\mathsf{DynFO}(\leq,+,\times)$ bound was known even in planar graphs under single edge changes.

▶ **Theorem 3.** BMMSize *is in* $\mathsf{DynFO}(\leq,+,\times)$ *under* $O(\frac{\log n}{\log\log n})$ *edge changes on classes of graphs for which non-zero circulation weights can be computed in* NC*.*

Main Technical Contributions. There are two major technical contributions of this work:

- *Converting the statically computed non-zero circulation weights for bipartite matchings to dynamically isolating weights for bipartite matchings.* Our main approach (described in detail in Section 3) is to assign polynomially bounded *isolating* weights to the edges of an evolving graph so that the minimum weight solution under these weights is *unique*. While static non-zero circulation weights guarantee this under deletions, for insertions, the dynamization is based on the seminal work of [15]. They construct isolating weights for perfect matching for arbitrary bipartite graphs, but which are quasipolynomially

Table 1 Previously known and new results in graphs with non-zero circulation weights in NC.

Problem	#changes		
	$O(1)$	$O(\frac{\log n}{\log\log n})$	$\log^{O(1)} n$
Reach	DynFO [8]	DynFO [13]	DynFO[⊕] [11]
Distance	DynFO[⊕]	DynFO[⊕]	DynFO[⊕]
BMMSize	DynFO	DynFO	DynFO[⊕]
BMMSearch	DynFO[⊕]	DynFO[⊕]	DynFO[⊕]

large in the size of the graphs. By assigning such weights only to the changed part of the graph and carefully combining with the previously assigned weights, we make sure the edge weights remain small as well as isolating throughout, using the Muddling Lemma (see Section 2). Our construction parallels that of [11] where dynamic isolating weights for reachability in non-zero circulation graphs were constructed based on the static construction from [24]. In addition to extending the reachability result (Theorem 1) this also enables us to prove a DynFO$(\leq,+,\times)$ bound (Theorem 3) for bipartite maximum matching (previously, a rather straightforward application of non-zero circulation weights in planar graphs could only achieve a DynFO[⊕] bound under single edge changes [26]).

- *Maintaining rank of a matrix under sublogarithmically many changes.* This involves non-trivially extending the technique from [8], which maintains rank under single entry changes, and combining it with [13] which shows how to compute the determinant of a small matrix of dimension $O(\frac{\log n}{\log\log n})$ in FO$(\leq,+,\times)$.

Organization. After some preliminaries in Section 2, in Section 3 we discuss the connection between dynamic isolation and static non-zero circulation and show its applications for matching and distance in Section 4 and Section 6, respectively. In Section 5, we describe the DynFO$(\leq,+,\times)$ algorithm for maximum matching, which is built on the rank algorithm under bulk changes from Section 7. Finally, we conclude with Section 8.

2 Preliminaries and Notations

Dynamic Complexity. The goal of a dynamic program is to answer a given query on an *input structure* subjected to insertion or deletion of tuples. The program may use an *auxiliary data structure* over the same domain. Initially, both input and auxiliary structures are empty; and the domain is fixed during each run of the program.

For a (relational) structure $\mathcal{I}$ over domain D and schema σ, a change $\Delta\mathcal{I}$ consists of sets R^+ and R^- of tuples for each relation symbol $R \in \sigma$. The result $\mathcal{I} + \Delta\mathcal{I}$ is the input structure where $R^{\mathcal{I}}$ is changed to $(R^{\mathcal{I}} \cup R^+) \setminus R^-$. The set of affected elements is the (active) domain of tuples in $\Delta\mathcal{I}$. A dynamic program $\mathcal{P}$ is a set of first-order formulas specifying how auxiliary relations are updated after a change. For a state $\mathcal{S} = (\mathcal{I}, \mathcal{A})$ with input structure $\mathcal{I}$ and auxiliary structure $\mathcal{A}$ we denote the state of the program after applying a change sequence α and updating the auxiliary relations accordingly by $\mathcal{P}_\alpha(\mathcal{S})$.

The dynamic program maintains a q-ary query Q under changes that affect k elements if it has a q-ary auxiliary relation ANS that at each point stores the result of Q applied to the current input structure. I.e., for each non-empty sequence α of changes affecting k elements, the relation ANS in $\mathcal{P}_\alpha(\mathcal{S}_\emptyset)$ and the relation $Q(\alpha(\mathcal{I}_\emptyset))$ coincide, where the state $\mathcal{S}_\emptyset = (\mathcal{I}_\emptyset, \mathcal{A}_\emptyset)$ consists of an input structure $\mathcal{I}_\emptyset$ and an auxiliary structure $\mathcal{A}_\emptyset$ over some common domain that both have empty relations, and $\alpha(\mathcal{I}_\emptyset)$ is the input structure after applying α.

If a dynamic program maintains a query, we say that the query is in DynFO. Similar to DynFO, one can define the class of queries DynFO$(\leq, +, \times)$ that allows for auxiliary relations initialized as a linear order, and the corresponding addition and multiplication relations. One can further extend this class by allowing parity quantifiers to yield the class DynFO$[\oplus](\leq, +, \times)$ and majority quantifiers to yield the class DynFO[MAJ]$(\leq, +, \times)$. The parity and majority functions of n bits $a_1, \ldots, a_n$ are true if $\sum_{i=1}^{n} a_i = 1 \pmod 2$ and $\sum_{i=1}^{n} a_i \geq n/2$, respectively. As we focus on changes of non-constant size, we include arithmetic in our setting. See [13, 11] for more details.

The *Muddling Lemma* [11] states that to maintain many natural queries, it is enough to maintain the query for a bounded number of steps, that we crucially use in this paper. In the following, we first recall the necessary notions before stating the lemma.

A query Q is *almost domain-independent* if there is a $c \in \mathbb{N}$ such that $Q(\mathcal{A})[(\text{adom}(\mathcal{A}) \cup B)] = Q(\mathcal{A}[(\text{adom}(\mathcal{A}) \cup B)])$ for all structures $\mathcal{A}$ and sets $B \subseteq A \setminus \text{adom}(\mathcal{A})$ with $|B| \geq c$. Here, $\text{adom}(\mathcal{A})$ denotes the *active domain*, the set of elements that are used in some tuple of $\mathcal{A}$. A query Q is $(\mathcal{C}, f)$*-maintainable*, for some complexity class $\mathcal{C}$ and some function $f : \mathbb{N} \to \mathbb{R}$, if there is a dynamic program $\mathcal{P}$ and a $\mathcal{C}$-algorithm $\mathbb{A}$ such that for each input structure $\mathcal{I}$ over a domain of size n, each linear order $\leq$ on the domain, and each change sequence α of length $|\alpha| \leq f(n)$, the relation Q in $\mathcal{P}_\alpha(\mathcal{S})$ and $Q(\alpha(\mathcal{I}))$ coincide, where $\mathcal{S} = (\mathcal{I}, \mathbb{A}(\mathcal{I}, \leq))$. AC^i is the class of problems that can be solved using polynomial-size circuit of $O(\log^i n)$ depth and NC $= \cup_i \mathsf{AC}^i$.

▶ **Lemma 4** (Muddling Lemma [11]). *Let Q be an almost domain independent query, and let $c \in \mathbb{N}$ be arbitrary. If the query Q is* $(\mathsf{AC}^d, 1)$*-maintainable under changes of size* $\log^{c+d} n$ *for some $d \in \mathbb{N}$, then Q is in* DynFO$(\leq, +, \times)$ *under changes of size* $\log^c n$.

There are several roughly equivalent ways to view the complexity class DynFO as capturing:

- The dynamic complexity of maintaining a Pure SQL database under fixed (first-order) updates and queries (the original formulation from [28]).
- The circuit dynamic complexity of maintaining a property where the updates and queries use uniform AC^0 circuits (see [2] for the equivalence of uniform AC^0 and FO).
- The parallel dynamic complexity of maintaining a property where the updates and queries use constant time on a CRCW PRAM (for the definition of Concurrent RAM, see [23]).

The first characterization is popular in the Logic and Database community, while the second is common in more complexity-theoretic contexts. The third one is useful to compare and contrast this class with dynamic algorithms, which essentially classify dynamic problems in terms of the sequential time for updates and queries. Operationally, our procedure is easiest to view in terms of the second or even the third viewpoint. We would like to emphasize that modulo finer variations based on built-in predicates (like arithmetic and order) in the first variation, uniformity in the second one and built-in predicates (like shift) in the third one, the three viewpoints are entirely equivalent.

We refer the readers to [8, 13, 11] for more discussion on the basics of the dynamic complexity framework.

Weight function and Circulation. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. By $G = (V, \vec{E})$ denote the corresponding graph where each of its edges is replaced by two directed edges, pointing in opposite directions. Let $|V| = n$ and we use the natural interpretation of the universe i.e., the set of vertices as the natural numbers from $[n]$.

A set system $\mathcal{M}$ on a universe U is a family of subsets of U i.e. $\mathcal{M} \subseteq 2^U$. Examples include the family of s,t-shortest paths and perfect matchings in a graph. A weight function $w : U \to \mathbb{Z}_{\geq 0}$ (the set of non-negative integers) induces a weight of $w(M) = \sum_{e \in M} w(e)$ on an element $M \in \mathcal{M}$. Such a weight function is said to be *isolating* for $\mathcal{M}$ if there is at most one element $M_0 \in \mathcal{M}$ with the minimum weight. The notion of isolation can be extended to a collection of families of graphs such as the collection of families of s,t-shortest paths for all $s,t \in V$. The weight function $w(e) = 2^{(n+1)u+v}$ for $e = \{u,v\}$, $u < v$, is a trivial isolating function – instead we want to give weights polynomially bounded in the size of the universe. A randomized construction of such a weight function is known for arbitrary set systems [27].

A function $w : \overleftrightarrow{E} \to \mathbb{Z}$ is called *skew-symmetric* if for all $e \in \overleftrightarrow{E}$, $w(e) = -w(e^r)$ (where e^r represent the edge with its direction reversed). The *circulation* of a directed cycle under a skew symmetric weight function is the absolute value of the sum of weights of the directed edges in the cycle. The skew-symmetric weight function w induces a *non-zero circulation* on the graph if every directed cycle in the graph gets a non-zero circulation under w.

We know from [3] that if w assigns non-zero circulation to every cycle that consists of edges of $\overleftrightarrow{E}$, then it isolates a directed path between each pair of vertices in $G = (V, \overleftrightarrow{E})$. Also, if G is a bipartite graph, then the weight function w can be used to construct a weight function $w' : E \to \mathbb{Z}$ that isolates a perfect matching in G [29]. For planar [29], bounded genus [10], bounded treewidth [11], and for any single crossing minor-free graph [4] non-zero circulation weights can be computed *deterministically* in Logspace, which is a subclass of NC.

Our convention represents by $\langle w_1, \ldots, w_k \rangle$ the weight function that on edge e takes weight $\sum_{i=1}^{k} w_i(e) B^{k-i}$, where $w_1, \ldots, w_k$ are weight functions such that $\max_{i=1}^{k} (n \cdot w_i(e)) \leq B$.

Maintaining Witnesses. The proof of [27] for the construction of a perfect matching witness carries over to the dynamic setting also and allow us to maintain a witness to the solution in DynFO[$\oplus$]($\leq, +, \times$). Since the perfect matching is isolated, from its weight one can infer the edges in the matching by deleting the edges one a time in parallel and see if the weight remains unchanged and accordingly place the edge in the matching. This is doable in FO[$\oplus$]($\leq, +, \times$). The extraction procedure for shortest path and maximum cardinality matching is similar.

3 Dynamic Isolation from Static Non-Zero Circulation

We know (from Section 2) that non-zero circulation weights are isolating weights. Thus, statically (when the given graph does not change over time) we can use them to obtain efficient parallel algorithm for distance and matching. However maintaining these weights seems to be hard in an evolving graph. This is because even for planar graphs where static non-zero circulation weights are easy to construct [3, 9, 29] maintaining them dynamically seems to need a dynamic planar embedding algorithm. And that alone doesn't seem to suffice since even small changes in the input can lead to large changes in the embedding and that will require us to change the weights of many edges (since weight of the edges are determined by the embedding [3, 29]). This induces us to side-step maintaining non-zero circulation weights.

In this paper, we circumvent this problem by modifying the approach of [15] to convert the given static nonzero-circulation weights to dynamic isolating weights. Notice that [15] yields a black box recipe to produce isolating weights of quasipolynomial magnitude for bipartite graphs, which we label as *FGT-weights* in the following way. Let $e_1, e_2, \ldots$ be the edges of a bipartite graph. Consider a non-zero circulation of exponential magnitude viz. $w_0(e_i) = 2^i$. Next, consider a list of $\ell = O(\log n)$ primes $\vec{p} = (p_1, \ldots, p_\ell)$, each of $O(\log n)$ bits, which yield a weight function $w_{\vec{p}}(e_i)$. This is defined by taking the ℓ weight functions $w_0 \bmod p_j$ for $j \in \{1, \ldots, \ell\}$ and concatenating them with *shifting* the weights to the higher-order bits appropriately, that is: $w_{\vec{p}}(e) = \langle w_0(e) \bmod p_1, \ldots, w_0(e) \bmod p_\ell \rangle$. This is so that there is no overflow from the j-th to the $(j-1)$-th weight for any $j \in \{2, \ldots, \ell\}$. In [15], it was proved that for every graph there exist some set of ℓ primes such that the respective weight function $w_{\vec{p}}$ isolates a perfect matching in the graph.

Suppose we start with a graph with static polynomially bounded weights ensuring non-zero circulation. In a step, some edges are inserted or deleted. The graph after deletion is a subgraph of the original graph; hence the non-zero circulation remains non-zero after a deletion[1], but we have to do more in the case of insertions. We aim to give the newly inserted edges FGT-weights in the higher-order bits while giving weight 0 to all the original edges in G again in the higher-order bits. Thus the weight of all perfect matchings that survive the deletions in a step remains unchanged. Moreover, if none such survive but new perfect matchings are introduced (due to insertion of edges) the lightest of them is determined solely by the weights of the newly introduced edges. In this case, our modification of the existential proof from [15] ensures that the minimum weight perfect matching is unique.

In order to handle bulk insertion of $N = \log^{O(1)} n$ edges, we need to apply the FGT-recipe described above to a set system with a universe of N elements. This yields quasipolynomial ($N^{\log^{O(1)} N}$) weights in N which are therefore still subpolynomial in n ($2^{(\log\log n)^{O(1)}} = 2^{o(\log n)} = n^{o(1)}$). Further, the number of primes is polyloglog ($\log^{O(1)} N = (\log\log n)^{O(1)}$) and so sublogarithmic ($\log^{o(1)} n$). Hence, the number of possible different weights is subpolynomial, which allows us to derandomize our algorithm by going over all possible FGT-weight functions defined above. We point out that in [11] a similar scheme is used for reachability and bears the same relation to [24] as this section does to [15]. We have the following lemma, which we prove in Section 3.1. Our proof of the lemma is crucially based on [15] but our proof is self contained except for Lemma 9 which we assume as a black box.

▶ **Lemma 5.** *Let G be a bipartite graph with non-zero circulation w^{old}. Suppose $N = \log^{O(1)} n$ edges are inserted into G to yield G^{new}. Then we can compute polynomially many weight functions in $\mathsf{FO}(\leq, +, \times)$ that have $O(\log n)$ bit weights, and at least one of them, w^{new} is isolating. Furthermore, the weights of the original edges remain unchanged under w^{new}.*

3.1 Details of Maintaining Dynamic Isolating Weights

We divide the edges of the graph into *real* and *fictitious*, where the former represents the newly inserted edges and the latter original undeleted edges[2].

Next, we follow the proof idea of [15] but focus on assigning weights to only real edges which are $N = \log^{O(1)} n$ in number. We do this in $\log N$ stages starting with a graph $G_0 = G$ and ending with an acyclic graph G_ℓ that contains a unique perfect matching if and only if G contains a perfect matching, where $\ell = \log N$. At each step, we maintain the following.

▶ **Invariant 6.** For $i \geq 1$, G_i contains no cycles with at most 2^{i+1} real edges.

Assuming this invariant we complete the proof of Lemma 5:

Proof of Lemma 5. From the invariant above G_ℓ does not contain any cycle that consists of real edges. From the construction of G_ℓ, if G has a perfect matching, then so does G_ℓ and hence it is a perfect matching. Notice that W_ℓ is obtained from $p_1, \ldots, p_\ell$ that include $O((\log\log n)^2) = o(\log n)$ many bits. Thus there are (sub)polynomially many such weighting functions W_ℓ, depending on the primes $\vec{p}$. Let $w = B \cdot W_\ell + w^{old}$ where we recall that $W_\ell(e)$

[1] If we merely had isolating weights, this would not necessarily preserve isolation.
[2] We use the terms old ↔ fictitious and new ↔ real interchangeably in this section.

is non-zero only for the new (real) edges and w^{old} is non-zero only for the old (fictitious) edges. Thus, any perfect matching that consists of only old edges is lighter than any perfect matching containing at least one new edge. Moreover, if the real edges in two matchings differ, then, from the construction of W_ℓ (for some choice of $\vec{p}$) both matchings cannot be the lightest as W_ℓ real isolates a matching. Thus the only remaining case is that we have two distinct lightest perfect matchings, which differ only in the old edges. But the symmetric difference of any two such perfect matchings is a collection of cycles consisting of old edges. But each cycle has a non-zero circulation in the old graph and so we can obtain a matching of even lesser weight by replacing the edges of one of the matchings in one cycle by the edges of the other one. This contradicts that both matchings were of least weight. ◀

Next we show how Invariant 6 is maintained. Notice that the case $i = 0$ follows from the above discussion and the induction starts at $i > 0$.

We first show how to construct G_{i+1} from G_i such that if G_i satisfies the inductive invariant 6, then so does G_{i+1}. In the i-th step, let $\mathcal{C}_{i+1}$ be the set of cycles that contain at most 2^{i+2} real edges, for $i > 1$. For each such cycle $C = f_0, f_1, \ldots$ containing $k \leq 2^{i+2}$ real edges (with f_0 being the least numbered real edge in the cycle), edge-partition the cycle into 4 consecutive paths $P_j(C)$ for $j \in \{0, 1, 2, 3\}$ such that the first three paths contain exactly $\lfloor \frac{k}{4} \rfloor$ real edges and the last path contains the rest. In addition ensure that the first edge in each path is a real edge. Let the first edge of the 4-paths be respectively $f'_0(= f_0), f'_1, f'_2, f'_3$. We identify each cycle in $\mathcal{C}_{i+1}$ with these 4-tuples $\langle f'_0, f'_1, f'_2, f'_3\rangle$.

For a cycle $C \in \mathcal{C}_{i+1}$, we define a set C' which consists of only real edges of C. Similarly, $\mathcal{C}'_{i+1} = \{C' \mid C \in \mathcal{C}_{i+1} \wedge C' \neq \emptyset\}$. Note that there can be many cycles in $\mathcal{C}_{i+1}$ corresponding to a single set in $\mathcal{C}'_{i+1}$ (i.e., those cycles that contain the same set of real edges). We fix a particular $C \in \mathcal{C}_{i+1}$ for every $C' \in \mathcal{C}'_{i+1}$ with which it is associated. The 4-tuple associated with the cycle C is also associated with the corresponding set C'. We have the following which shows that the associated 4-tuples $\langle f'_0, f'_1, f'_2, f'_3\rangle$ uniquely characterise sets in $\mathcal{C}'_{i+1}$.

▷ Claim 7. There is at most one set in $\mathcal{C}'_{i+1}$ that has a given 4-tuple $\langle f'_0, f'_1, f'_2, f'_3\rangle$ associated with it.

Proof. Suppose two distinct sets $C'_1, C'_2 \in \mathcal{C}'_{i+1}$ have a common 4-tuple $\langle f'_0, f'_1, f'_2, f'_3\rangle$ associated with them. Let C_1, C_2 be two cycles corresponding to C'_1 and C'_2, respectively. Then for at least one $j \in \{0, 1, 2, 3\}$, $P_j(C_1) \neq P_j(C_2)$. Hence, $P_j(C_1) \cup P_j(C_2)$ is a closed walk in G_i containing at most $2 \times \lceil \frac{2^{i+2}}{4} \rceil = 2^{i+1}$ real edges, contradicting the assumption on G_i. ◁

This shows that there are at most N^4 elements in $\mathcal{C}'_{i+1}$ because that is the maximum number of distinct 4-tuples of real edges. We define circulation for the sets in $\mathcal{C}'_{i+1}$ via the circulation for those in $\mathcal{C}_{i+1}$. We know that for every $C' \in \mathcal{C}'_{i+1}$ there is at least one $C \in \mathcal{C}_{i+1}$ corresponding to it. Let w be a weight function that gives non-zero weights to only real edges of the graph. Circulation of $C' \in \mathcal{C}'_{i+1}$ with respect to w is defined as $c_w(C') = c_w(C) = |w(e_1) - w(e_2) + w(e_3) - \ldots|$, where $e_i \in C$. This is well-defined since in bipartite graphs the parity of the length of any two paths between the same pair of vertices is the same. Thus, all C such that they have C' associated with it, have the same circulation since the sign associated with a real edge does not change for any such C. Now we will use the following lemma to ensure non-zero circulations to sets in $\mathcal{C}'_{i+1}$.

▶ **Lemma 8** (Based on Lemma 2 in [16]). *For every constant $k > 0$ there is a constant $k_0 > 0$ such that for every set S of m-bit integers with $|S| \leq m^k$, the following holds: There is a $k_0 \log m$-bit prime number p such that for any $x, y \in S$, if $x \neq y$ then $x \not\equiv y \bmod p$.*

We apply the above lemma to the set $c_{w_0}(\mathcal{C}'_{i+1}) = \{c_{w_0}(C') : C' \in \mathcal{C}'_{i+1}\}$. Here, the weight function w_0 assigns weights $w_0(e_j) = 2^j$ to the real edges which are $e_1, e_2, \ldots, e_N$ in an arbitrary but fixed order. Notice that from the above claim, the size of this set is $|\mathcal{C}'_{i+1}| \leq N^4$. Since $w_0(e_j)$ is j-bits long hence $c_{w_0}(C)$ for any cycle $C \in \mathcal{C}_i$ that has less than 2^{i+2} real edges is at most $i + j + 2 < 4N$-bits long. Thus, we obtain a prime p_{i+1} of length at most $k_0 \log(4N)$ by picking $k = 4$. We define $w_{i+1}(e_j) = w_0(e_j) \bmod p_{i+1}$. By Lemma 8 we know that circulation of each set in $\mathcal{C}'_{i+1}$ is non-zero with respect to w_{i+1}. Therefore, circulation of all the cycles in $\mathcal{C}_{i+1}$ is nonzero with respect to w_{i+1} (remember that w_{i+1} assign nonzero weights to only real edges).

Now consider the following crucial lemma from [15]:

▶ **Lemma 9** ([15]). *Let $G = (V, E)$ be a bipartite graph with a weight function w. Let C be a cycle in G such that $c_w(C) \neq 0$. Let E_1 be the union of all minimum weight perfect matchings in G. Then the graph $G_1 = (V, E_1)$ does not contain the cycle C. Moreover, all the perfect matchings in G_1 have the same weight.*

Let B be a large enough constant (though bounded by a polynomial in N) to be specified later. We shift the original accumulated weight function W_i and add the new weight function w_{i+1} to obtain: $W_{i+1}(e) = W_i(e)B + w_{i+1}(e)$. Apply W_{i+1} on the graph G_i to obtain the graph G_{i+1}. Inductively suppose we have the invariant 6 that the graph G_i did not have any cycles containing at most 2^{i+1} real edges. This property is preserved when we take all the perfect matchings in G_i and apply W_{i+1} yielding G_{i+1}. Moreover, from Lemma 9 and the construction of w_{i+1}, the cycles of $\mathcal{C}_i$ disappear from G_{i+1} restoring the invariant. This yields a weight function W_ℓ using that $\ell = \log n$ (see the discussion before Invariant 6).

Notice that it suffices to take B greater than the number of real edges times the maximum of $w_i(e)$ over i, e. Showing that G_1 contains no cycle with at most 4 real edges mimics the above more general proof, and we skip it here. We say that a weight function that gives non-zero weights to the real edges, *real isolates* $\mathcal{M}$ for a set system $\mathcal{M}$ if the minimum weight set in $\mathcal{M}$ is unique with respect to that weight function. In our context, $\mathcal{M}$ will refer to the set of perfect/maximum matchings.

4 Maximum Cardinality Matching Search in DynFO$[\oplus](\leq, +, \times)$

In this section, we convert the static algorithm for maximum matching search in bipartite graphs into a dynamic algorithm with the help of the isolating weights from the previous section. In the static setting [6] the problem reduces to determining non-singularity of an associated matrix given a non-zero circulation for the graph.

The algorithm extracts what is called a min-weight *generalized* perfect matching (min-weight GPM), that is, a matching along with some self-loops. The construction proceeds by adding a distinct edge (v, t_v) on every vertex $v \in V(G)$ with a self-loop on the new vertex t_v to yield the graph G'. The idea is to match as many vertices as possible in G' using the actual edges of G while reserving the pendant edges (v, t_v) to match vertices that are unmatched by the maximum matching. If a vertex v is matched in a maximum matching of G then the vertex t_v is "matched" using a self-loop.

Given a non-zero circulation weight w''' for G the weight function for G' is $w = \langle w', w'', w''' \rangle$. Here we represent by w' the function that is identically 0 for all the self loops and is 1 for all the other edges. $w''(e)$ is zero except for pendant edges $e = (v, t_v)$, for $v \in V(G)$, which have $w''(e) = v$ (where v is interpreted as a number in $\{1, \ldots, |V(G)|\}$) such that all vertices get distinct numbers. The paper [6] considers the weighted Tutte matrix

T where for an edge (u, v) the entry $T(u, v) = \pm x^{w(u,v)}$ (say, with a positive sign iff $u < v$) and is zero otherwise. It shows that in the univariate determinant polynomial $det(T)$ the least degree term x^W with a non-zero coefficient must have this coefficient equal to ± 1 and the exponent W is the weight of the minimum weight generalized perfect matching in G'. Further this min-weight GPM consists of a maximum cardinality matching in G along with the edges (v, t_v) for all the vertices v unmatched in the maximum matching. The edges in the maximum matching can then be obtained by checking if, on removing the edge (u, v), the weight of the min-weight maximum matching increases.

The idea behind the proof is as follows:

1. The most significant weight function w' ensures that the cardinality of the actual edges (i.e. edges from G) picked in the min-wight GPM in G' equals the cardinality of the maximum matching in G. This is because the GPM would cover as many of the t_v vertices with self-loops as possible to minimize the weight that ensures the corresponding v must be covered by an actual edge.
2. The next most significant weight function w'' is used to ensure that all the min-weight GPMs use the same set of pendant edges. This is because, otherwise, there is an alternating path in the symmetric difference of the two GPMs that starts and ends at self-loops t_u, t_v. Then, the difference in the weights w'' of the two matchings restricted to the path is $u - v \neq 0$ and we can find a GPM of strictly smaller weight by replacing the edges of one matching with the edges of the other matching restricted to the path, contradicting that both matchings were the lightest GPMs.
3. The least significant weight function w''' then isolates the GPM since all min-weight GPMs are essentially perfect matchings restricted to the same set S of vertices, namely those that are not matched by the corresponding pendant edges and the non-zero circulation weights on G ensures that these are isolating weights on the induced graph $G[S]$.

We claim that we just need *isolating* weights w''' instead of non-zero circulation weights to ensure that the above technique works. Replacing non-zero circulations with isolating weights does not affect the first two steps. It would seem, the third step does not work since isolating weights for G might not be isolating weights for the subgraph $G[S]$. However, Lemma 5 can be applied to the graph $G[S]$ directly – notice that in the above proof sketch S is determined by the first two weight functions w', w'' and does not depend on the third w'''.

As described above, we need to maintain the determinant of a certain matrix A related to the Tutte matrix in order to find the size of the maximum cardinality matching. For a small change matrix B, the Matrix Determinant Lemma [30, 17]

$$\det(A + UBV) = \det(I + BVA^{-1}U)\det(A)$$

allows us to maintain the determinant by reducing it to maintaining the inverse of the matrix. To maintain the inverse, the Sherman-Morrison-Woodbury formula [19]

$$(A + UBV)^{-1} = A^{-1} - A^{-1}U(I + BVA^{-1}U)^{-1}BVA^{-1}$$

tells us how the task of recomputing the inverse of a non-singular matrix A under small changes B reduces to that of computing the inverse of a small matrix $(I+BVA^{-1}U)$ statically. So we need to ensure that the matrix remains invertible throughout which is what we achieve below by tinkering with the definition of the Tutte matrix. We have the following definition:

▶ **Definition 10.** *The generalized Tutte matrix is a matrix with rows and columns indexed by $V(G') = V(G) \cup \{t_v : v \in V(G)\}$ and with the following weights on edges: $T(t_v, t_v) = 1$, $T(v, v) = x^{w_\infty}$ and $T(a, b) = \pm x^{\langle w', w'', w''' \rangle(a,b)}$ whenever $a, b \in V(G'), a \neq b$. It is ensured in the above that $T(a, b) = -T(b, a)$. Here w_∞ is a polynomially bounded number larger than the largest of the weights $\langle w', w'', w''' \rangle (a, b)$.*

Note that the generalized Tutte matrix is not unique. We now have the following:

▶ **Lemma 11.** *Let T be a generalized Tutte matrix defined above, then the highest exponent w such that x^w divides $\det(T)$ is twice the weight of the min-weight GPM in G'. Further, the matrix T is invertible.*

Proof. From the properties of the weight function $\langle w', w'', w''' \rangle$ we see that the minimum weight generalized perfect matching is unique see [6, Lemma 9]. The exponent of the least degree monomial is twice the weight of the min-weight GPM when we take superposition of the unique min-weight GPM with itself [6, Observation 14]. The self loops on the vertices t_v appear only once in a monomial and twice in the superposition, but because their weight is zero under $\langle w', w'', w''' \rangle$ it will not affect the exponent.

In order to guarantee invertibility of T we just need to prove that the product of the diagonal terms yields a monomial of much higher degree than any other monomial and of coefficient one, since this implies that the matrix is non-singular because this monomial cannot be canceled out by the rest of the monomials. Consider the product of the diagonal entries viz. $x^{\frac{|V(G')|}{2} w_\infty} = x^{|V(G)| w_\infty}$. Any monomial with less than $|V(G)|$ diagonal entries $T(v, v)$ is bound to be of much smaller exponent. Now the monomials which use an off-diagonal entry $T(v, u)$ or $T(v, t_v)$ must miss out on the diagonal entry in the v-th row, making the exponent much smaller. ◀

4.1 Maintaining the Determinant and Inverse of a Matrix

We need the following definitions and results about univariate polynomials, matrices of univariate polynomials and operations therein over a finite field of characteristic 2. Let $\mathbb{F}_2$ be the field of characteristic 2 containing 2 elements. For (potentially infinite) power series $f, g \in \mathbb{F}_2[[x]]$, we say f *m-approximates* g (denoted by $f \approx_m g$) if the first m terms of f and g are the same. We will extend this notation to matrices and write $F \approx_m G$ where $F, G \in \mathbb{F}_2[[x]]^{\ell \times \ell}$ are matrices of power series. We will have occasion to use this notation only when one of F, G is a matrix of polynomials, that is, a matrix of finite power series.

Notice that if $A \in \mathbb{F}_2[x]^{n \times n}$ with the degree of entries bounded by w_∞, then there exists $A^{-1} \in \mathbb{F}_2[[x]]^{n \times n}$. For us, only the monomials with degrees at most w_∞ are relevant. Thus we will assume that we truncate A^{-1} at w_∞ many terms to yield matrix $A' \approx_{w_\infty} A^{-1}$. Then we have the following:

▶ **Lemma 12** (Lemma 10 in [11])**.** *Suppose $A \in \mathbb{F}_2[x]^{n \times n}$ is invertible over $\mathbb{F}_2[[x]]$, and $C \in \mathbb{F}_2[x]^{n \times n}$ is an m-approximation of A^{-1}. If $A + UBV$ is invertible over $\mathbb{F}_2[[x]]$ with $U \in \mathbb{F}_2[x]^{n \times \ell}, B \in \mathbb{F}_2[x]^{\ell \times \ell}$, and $V \in \mathbb{F}_2[x]^{\ell \times n}$, then $(A + UBV)^{-1} \approx_m C - CU(I + BVCU)^{-1}BVC$. Furthermore, if $\ell \leq \log^c n$ for some fixed c and all involved polynomials have polynomial degree in n, then the right-hand side can be computed in $\mathsf{FO}[\oplus](\leq, +, \times)$ from C and ΔA.*

Similar to the above (using the closure of m-approximation under product), we get an approximate version of the Matrix Determinant Lemma, that is:

▶ **Proposition 13.** *Suppose $A \in \mathbb{F}_2[x]^{n \times n}$ is invertible over $\mathbb{F}_2[[x]]$, and $C \in \mathbb{F}_2[x]^{n \times n}$ is an m-approximation of A^{-1} and polynomial $d(x) \approx_m \det(A)$ then $d \cdot \det(I + BVCU) \approx_m \det(A + UBV)$.*

We can now proceed to prove Theorem 1:

Proof of Theorem 1. By putting $m = w_\infty$ and applying Lemma 12 and Propositions 13 to the generalized Tutte matrix from Lemma 11 and using the (Muddling) Lemma 4, we complete the matching part of Theorem 1 (see Section 6 for the proof involving distance). ◀

5 Maximum Cardinality Matching in DynFO$(\leq, +, \times)$

In this section we prove Theorem 3 by giving a DynFO$(\leq, +, \times)$ algorithm for maintaining the size of a maximum matching under $O(\frac{\log n}{\log\log n})$ changes. Notice that the approach in the previous section has the limitation that it only gives a DynFO$[\oplus](\leq, +, \times)$ bound as we need to maintain polynomials of large (polynomial in n) degree. Instead, the main ingredient here is a new algorithm for maintaining the rank of a matrix in DynFO$(\leq, +, \times)$ under $O(\frac{\log n}{\log\log n})$ changes (Section 7). Our matching algorithm follows the basic approach of the non-uniform DynFO algorithm of [8]. Here, since we use deterministic isolation weights (as opposed to the randomized isolation weights of [27]), with some more work, we obtain a *uniform* DynFO$(\leq, +, \times)$ bound under bulk changes.

The algorithm of [8] builds on the well-known correspondence between the size of maximum matching and the rank of the Tutte matrix of the corresponding graph − if a graph contains a maximum matching of size m then the associated Tutte matrix is of rank $2m$ [25]. The dynamic rank algorithm from Section 7 cannot be applied directly since the entries of the Tutte matrix are indeterminates. However, the rank can be determined by replacing each x_{ij} by $2^{w(i,j)}$. Here w assigns a positive integer weight to every edge (i, j) under which the maximum matching gets unique minimal weight, i.e., it is matching-isolating. Using the Isolation Lemma [27], it can be shown that the correspondence between the rank and the size of the maximum matching does not change after such a weight transformation [22, 8].

Our algorithm diverges from [8] as we need to deterministically compute these isolating weights and also, to somehow maintain those. Since we do not know how to maintain such weights directly, as in Section 4, we convert the static non-zero circulation weights to dynamic isolating weights using the Muddling Lemma 4. Given a graph G, let B_w be its weighted Tutte matrix with each x_{ij} replaced by $2^{w(i,j)}$ for an isolating weight function w. Initially, the static non-zero circulation weights provide such weights. Since we are only interested in computing the rank of B_w, we do not need to make the initial modifications of adding pendant edges or self-loops to G as before. So the weight function w is just the non-zero circulation weight $\langle w''' \rangle$ here. In the dynamic process, similar to Section 4, we use the FGT-weights w_{new} on top for the newly inserted edges. We have the following:

▶ **Lemma 14.** *Given a dynamic algorithm for maintaining the rank of an integer matrix under $k = O(\log^c n)$ changes at each step for some fixed constant c, we can maintain the size of the maximum matching in the same complexity class under $O(k)$ changes for the class of graphs where non-zero circulation weights can be computed in* NC.

Proof. Given a graph G, assume we have an algorithm for computing the non-zero circulation weight function w in $\mathsf{NC}^i \subseteq \mathsf{AC}[\frac{\log^i n}{\log\log n}]$ for some fixed integer i. Once these weights w are available, rank(B_w) can be found in NC^2 [1] which is contained in $\mathsf{AC}[\frac{\log^2 n}{\log\log n}]$. Since $O(k)$ changes can occur at each step, during this time, total of $O(k \cdot (\frac{\log^i n}{\log\log n} + \frac{\log^2 n}{\log\log n}))$ many new changes accumulate. As w assigns non-zero circulation weights to the edges of G, we can assign weight 0 to the deleted edges and the weights remain isolating. For the newly inserted edges, which are only polylog(n) many, we compute the polynomially bounded FGT-weights in AC^0 using Lemma 9. Thanks to Lemma 4, in $O(\frac{\log^i n}{\log\log n} + \frac{\log^2 n}{\log\log n}))$ many steps we can take care of all the insertions by adding k new edges at each step along with k old ones in double the speed using our rank algorithm. Note that, during the static rank computation phase, we do not restart the static algorithm for computing the weight w. Instead, we recompute these weights once the rank computation using them finishes. More precisely, we can think of a combined static procedure that computes the non-zero circulation weights followed by the rank of the weighted Tutte matrix B_w in NC^b for $b=\max(i,2)$. And on this combined procedure, we apply our Muddling Lemma 4. ◀

We can now prove Theorem 3:

Proof of Theorem 3. Similar to [8, Theorem 16] this implies a *uniform* bounded expansion first-order truth-table (bfo-tt) reduction from maximum matching to rank in this special case.[3] Since $\mathsf{DynFO}(\leq,+,\times)$ is closed under bfo-tt reductions [8, Proposition 4] and dynamic rank maintenance is in $\mathsf{DynFO}(\leq,+,\times)$ under $O(\frac{\log n}{\log\log n})$ changes (Theorem 2), in classes of graphs where non-zero circulation weights can be computed in NC we have the result. ◀

6 Maintaining Distance under Bulk Changes

In this section, extending the reachability result of [11], we show that distances can be maintained in $\mathsf{DynFO}[\oplus](\leq,+,\times)$ under polylog(n) changes in classes of graphs where non-zero circulation can be computed in NC. We start with describing the reachability algorithm of [11] followed by the necessary modifications needed for maintaining distance information.

6.1 Outline of the Approach for Reachability

Let $G=(V,E)$ be the given n-node graph with an isolating weight assignment w. For a formal variable x, let the corresponding weighted adjacency matrix $A=A_{(G,w)}(x)$ be defined as follows: if $(u,v)\in E$, then $A[u,v]=x^{w(u,v)}$, and 0 otherwise. Consider the matrix $D=(I-A)^{-1}$, where I is the identity matrix. Notice that the matrix $(I-A)$ is invertible over the ring of formal power series (see [13]). Here $D=\sum_{i=0}^{\infty}(A)^i$ is a matrix of formal power series in x and in the (s,t)-entry, the coefficient of the i-th terms gives the number of walks from s to t of weight i.

As w isolates the minimal weight paths in G, it is enough to compute these coefficients modulo 2 for all i up to some polynomial in n since there is a unique path with the minimal weight if one exists. So, it is enough to compute and update the inverse of the matrix $I-A$. Though to do it effectively, we compute the *n-approximation* C of D, which is a matrix of formal polynomials that agrees with the entries of D up to degree $i\leq n$ terms. This precision is preserved by the matrix operations we use, see [13, Proposition 14].

When applying a change ΔG to G that affects k nodes, the associated matrix A is updated by adding a suitable change matrix ΔA with at most k non-zero rows and columns, and can therefore be decomposed into a product UBV of suitable matrices U,B, and V, where B is a $k\times k$ matrix. To update the inverse, we employ the Sherman-Morrison-Woodbury identity (cf. [19]), which gives a way to update the inverse when A is changed to $A+\Delta A$ as follows:

$$(A+\Delta A)^{-1}=(A+UBV)^{-1}=A^{-1}-A^{-1}U(I+BVA^{-1}U)^{-1}BVA^{-1}.$$

[3] Intuitively, bounded expansion first-order (bfo) reductions are first-order reductions such that each tuple in a relation and each constant of the input structure affects at most a constant number of tuples and constants in the output structure. A bfo-tt reduction bears the same relation to a bfo-reduction as a truth-table reduction bears to a many-one reduction. For a formal definition see [8, Section 3.2] where it was first formalized.

The right-hand side can be computed in $\mathsf{FO}[\oplus](\leq,+,\times)$ for $k = \log^{O(1)} n$ since modulo 2 computation of (1) multiplication and iterated addition of polynomials over $\mathbb{Z}$ and (2) computation of the inverse of $I + BVA^{-1}U$ which is also a $k \times k$ matrix is possible in $\mathsf{FO}[\oplus](\leq,+,\times)$ for (matrices of) polynomials with polynomial degree [18]. Finally, we need to assign weights to the changed edges as well so that the resulting weight assignment remains isolating. We show how to achieve this starting with non-zero circulation weights. Using [11, Theorem 5] we can assume that such a weight assignment is given, and that we only need to update the weights once.

Let u be skew-symmetric non-zero circulation weights for G and let n^k be the polynomial bound on the weights. Further, let w be the isolating weight assignment that gives weight $n^{k+2} + u(e)$ to each edge $e \in E$. During the AC^d initialization, we compute the weights u and w and an n^b-approximation matrix C of $(I - A_{(G,w)}(x))^{-1} \bmod 2$ for some constant b.

When changing G via a change ΔE with deletions E^- and insertions E^+, the algorithm proceeds as follows: To compute the isolating weights w^-, the non-zero circulation weights u^- for G^- are obtained from u by setting the weight of deleted edges $e \in E^-$ to 0. As u^- gives the same weight to all simple cycles in G^- as u gives to these cycles in G, it has non-zero circulation. To handle E^+ it can be shown that [11, Lemma 11] there is a FO-computable (from w, E^+ and the reachability information in G) family W' of polynomially many weight assignments such that $\exists w' \in W'$ isolating for $(V, E \setminus E^- \cup E^+)$.

Hence we need to maintain polynomially many different instances of the graph with different weight functions from W' such that in at least one of them the paths are isolated. The idea is that if there is an s-t-path using at least one inserted edge from E^+, then there is a unique minimal path among all s-t-paths that use at least one such edge, while ignoring the weight of the paths that is contributed by edges from E. The edge weights from E^+ are multiplied by a large polynomial to ensure that the combined weight assignment with the existing weights for edges in E remains isolating. Since the weights are constructed only for a graph with $N = \log^{O(1)} n$ many nodes, and although they are not polynomially bounded in N, they *are* in n. Please refer to [11, Section 6] for more details.

From the above discussion, to prove a similar bound for distances, it suffices to show that (1) after every polylog(n) changes, we can ensure the edge weights remain "shortest path-isolating" and (2) under such weights the distance can be updated in $\mathsf{FO}[\oplus](\leq,+,\times)$.

6.2 Dynamic Isolation of Shortest Paths

In the following, we first describe how the isolating weights for reachability can be modified to give weights for isolating shortest paths. Similar to maintaining reachability, our algorithm handles deletions and insertions differently. In case of deletion, we set the weight of the deleted edges $e \in E^-$ to 0 and due to the non-zero circulation weights, the weights remain isolating. For insertions, the idea is to do a *weight refinement* by shifting the original edge weights $w(e)$ (1 in case of unweighted graphs) to the highest order bits in the bit-representation, in the presence of other newly assigned weights to the edges.

We define a new weight functions $w^* = \langle w, w', u \rangle$ and assign these weights to the inserted edges $e \in E^+$. The existing edges E are not assigned any w' weight and all those bits remain zeroes. So we get a family of weight functions W^*. Here w is the polynomially-bounded original edge weights, w' is one of the (polynomially many) isolating weights from the family W' assigned to the newly added edges E^+ during the dynamic process, and u are the non-zero circulation weights that are computed statically. The combined weights w^* is FO-constructible from the weights w, w' and u as all involving numbers are $O(\log n)$ bits long (see [21, Theorem 5.1]). The correctness of the fact that these weights are indeed shortest path isolating follows from [11, Lemma 11] with the observation that since the original edge weights are shifted to the highest-order bits, the minimum weight path with these combined weights corresponds to the shortest path in the original graph.

The update algorithm for maintaining reachability can be extended to maintaining distances also [13]. Here, instead of checking only the non-zeroness of the (s, t)-entry in the polynomial matrix C, we compute the minimum degree term as well (with coefficient 1), which can be done in $\mathsf{FO}[\oplus](\leq, +, \times)$. By construction, the degree of a term in this polynomial is same as the weight of the corresponding walk under the dynamic isolating weights and applying an easy transformation gives us back the original weights, that is, the weight of the shortest path from s to t in G'. This proves the distance part of Theorem 1.

7 Maintaining Rank under Bulk Changes

In this section we prove Theorem 2. For ease of exposition, we build upon the algorithm as described in [7, Section 3.1]. Before going into the details of the proof, we start with defining some important notation, followed by our overall proof strategy. Let A be a $n \times n$ matrix over $\mathbb{Z}_p$, where $p = O(n^3)$ is a prime. Let K be the kernel of A. For a vector $v \in \mathbb{Z}_p^n$, we define $S(v) = \{i \in n \mid (Av)_i \neq 0\}$, where $(Av)_i$ denotes the ith coordinate of the vector Av. Let B be a basis of Z_p^n. A vector $v \in B$ is called *i-unique* with respect to A and B if $(Av)_i \neq 0$ and $(Aw)_i = 0$ for all other $w \in B$. A basis B is called A-good if all the vectors in $B - K$ are i-unique with respect to A and B. For a vector $v \in B - K$, the minimum i for which it is i-unique is called the principal component of v, denoted as $\mathrm{pc}(v)$.

Starting with an A-good basis B, and introducing a small number of changes to yield A' may lead to B losing its A-goodness. To restore this, we alter the matrix B in two phases to obtain an A'-good basis B'. The first phase involves identifying a full rank submatrix in A' corresponding to the changed entries, inverting it, and restoring the pc's of the columns of that full rank submatrix. In the second phase, we restore the pc's of the rest of vectors, which had lost the pc's either because of the changes or during Phase 1. The rough outline is similar to that of [7] but in order to handle non-constant changes we have to make non-trivial alterations and use efficient small matrix inversion from [13]. We have Theorem 2 from the following lemma, whose proof is provided in Section 7.1:

▶ **Lemma 15.** *Let $A, A' \in \mathbb{Z}_p^{n\times n}$ be two matrices such that A' differs from A in $O(\frac{\log n}{\log\log n})$ places. If B is an A-good basis then we can compute an A'-good basis B' in $\mathsf{FO}(\leq, +, \times)$.*

▶ **Proposition 16** ([8]). *Let $A \in \mathbb{Z}_p^{n\times n}$ and B an A-good basis of $\mathbb{Z}_p^n$. Then $rank(A) = n - |B \cap K|$ is the number of vectors in the basis that have a pc.*

▷ Claim 17. If the rank of an $n \times n$ matrix A is r then there exists a prime $p = O(max(n, \log N)^3)$ such that the rank of A over $\mathbb{Z}_p$ is also r, where N is the maximum absolute value the entries of the matrix A contain.

Proof. We know that if the rank of A is r then there exists a $r \times r$ submatrix A_s of A such that its determinant is nonzero. The value of this determinant is at most $n!N^n$, which can be represented by $O(n(\log n + \log N))$ many bits. Therefore, this determinant is divisible by at most $O(n(\log n + \log N))$ many primes. Thus by the prime number theorem, we can say that for a large enough n there exists a prime p of magnitude $O(max(n, \log N)^3)$ such that determinant of A_s is not divisible by p. ◁

Hence, to compute the rank of A, it is sufficient to compute the rank of the matrices (A mod p) for all primes p of size $O(max(n, \log N)^3)$ and take the maximum among them. Below we show how to maintain the rank of the matrix A mod p for a fixed prime p. We replicate the same procedure for all the primes in parallel.

A' is the matrix that is obtained by changing k many entries of A. Notice that if B is not A'-good basis, that means there are some vectors in $B - K'$ which are not i-unique with respect to B and A', where K' is the kernel of A'. A vector $w \in B - K'$ which was i-unique ($i \in [n]$) with respect to A and B may no longer be i-unique with respect to A' and B for the following two reasons, (i) $i \notin S'(w)$, (ii) there may be more than one vector w' such that $i \in S'(w')$. For a vector v, $S'(v)$ denotes the set of non-zero coordinates of the vector $A'v$. Below we give an AC^0 algorithm to construct an A'-good basis.

In several places we make use of the fact that sum and product of polylog(n) many numbers each with of polylog(n) bits can be computed in AC^0 [21, Theorem 5.1].

7.1 Construction of an A'-good basis

Let $k = O(\frac{\log n}{\log \log n})$ and $M = (A'B)_{R,C}$ be the $n \times n$ matrix where R is the set of rows and C is the set of columns of M. We know that $A'B$ differs from AB in a set R_0 of at most k many rows. Let $M_{R_0,*}$ be the matrix M restricted to the rows in the set R_0.

▷ Claim 18. There exists a prime $q = O(\log^3 n)$ such that the rank of $(M_{R_0,*} \bmod q)$ is equal to the rank of $(M_{R_0,*} \bmod p)$.

Proof. The proof follows from the proof of Claim 17. ◁

From the above claim, it follows that a row basis of $(M_{R_0,*} \bmod p)$ remains a row basis of $(M_{R_0,*} \bmod q)$ for some $O(\log \log n)$-bit prime q. Next, we have two constructive claims:

▷ Claim 19. A row basis R_1 of $(M_{R_0,*} \bmod q)$ can be found in AC^0.

Proof. Note that number of rows in R_0 are $O(\frac{\log n}{\log \log n})$; thus, the number of the subsets of the rows of R_1 are polynomially many (in n), and each row in the set R_0 can be indexed by $O(\log \log n)$ many bits. An element of $\mathbb{Z}_q$ can also be represented by $O(\log \log n)$ many bits. Therefore, for a fixed subset S of R_0, all the linear combinations of the rows of S can be represented by $O(\log n)$ many bits. We try all the linear combinations in parallel. Also, we do this for all the subsets in parallel. The subset with the maximum cardinality in which all the linear combinations result in non-zero values will be the maximum set of linearly independent rows in $(M_{R_0,*} \bmod q)$. ◁

▷ Claim 20. A column basis C_1 of $(M_{R_1,*} \bmod q)$ can be found in AC^0.

Proof. To find the maximum set of linearly independent columns in the matrix $M_{R_1,*}$ we just check in parallel if the rank of $M_{R_1,i}$ is greater than the rank of $M_{R_1,i-1}$ for all $i \in [m]$. Let $c_1, c_2 \dots c_m$ be the columns in the matrix $M_{R_1,*}$. Note that the set of columns c_i such that the rank of $M_{R_1,i}$ is more than the rank of $M_{R_1,i-1}$, form a maximum set of linearly independent columns in $M_{R_1,i}$. We can check this in AC^0. ◁

We are going to construct four matrices $D^{(1)}, E^{(1)}, D^{(2)}, E^{(2)}$ successively such that the product $B' = B \times D^{(1)} \times E^{(1)} \times D^{(2)} \times E^{(2)}$ is an A'-good basis. For this, we need to show that each column c_i of $A'B'$ is either an all zero-column or there exists a unique j such that the j-th entry of the column is non-zero. In other words, each column of B' is either i-unique or it is in the kernel of A'. We will show how to obtain each of the four matrices above as well as take their product in AC^0. We need a technical lemma before we start.

Combining Matrices. Here we state a lemma about constructing matrices from smaller matrices that we we will use several times. Let $X \in \mathbb{Z}_p^{n\times n}$ be a matrix and let $X^{1,1} \in \mathbb{Z}_p^{\ell\times\ell}$, $X^{1,2} \in \mathbb{Z}_p^{\ell\times(n-\ell)}$, $X^{2,1} \in \mathbb{Z}_p^{(n-\ell)\times\ell}$, $X^{2,2} \in \mathbb{Z}_p^{(n-\ell)\times(n-\ell)}$ be 4 matrices and let $R, C \subseteq [n]$ be two subsets of indices of cardinality ℓ each. Let $\bar{R} = [n] \setminus R, \bar{C} = [n] \setminus C$. Then we have:

▶ **Lemma 21.** *Given the matrices $X^{i,j}$ for $I, J \in [2]$ and the sets R, C explicitly for $|R| = |C| = \ell = (\log n)^{O(1)}$, we can construct, in AC^0, the matrix Y such that $Y_{R,C} = X^{1,1}$, $Y_{R,\bar{C}} = X^{1,2}$, $Y_{\bar{R},C} = X^{2,1}$ and $Y_{\bar{R},\bar{C}} = X^{2,2}$.*

Proof. Notice that the sets R, C can be sorted in AC^0 because computing the position $pos_R(r)$ of an element $r \in R$ (i.e., the number of elements not larger than r) is equivalent to finding the sum of at most ℓ bits (which are zero for elements of R larger than r and one otherwise).

The position $pos_{\bar{R}}(r')$ of an element $r' \in \bar{R}$ (i.e. the number of elements of $\bar{R}$ not larger than r') can also be found in AC^0. This is because we can first find the set $R(r') = \{r_i \in R : r_i < r'\}$ in AC^0. Then $pos_{\bar{R}}(r') = r' - |R(r')|$ because there are r' rows with indices at most r' and out of these all but $|R(r')|$ are in $\bar{R}$ and thus can be computed in AC^0. We can similarly compute $pos_C(c), pos_{\bar{C}}(c')$ for $c \in C$ and $c' \in \bar{C}$.

Finally given $i, j \in [n]$ the element $Y_{i,j}$ is $X^{1,1}_{pos_R(i),pos_C(j)}$ if $i \in R, j \in C$. Similarly if $i \in \bar{R}, j \in C$ then it is $X^{2,1}_{pos_{\bar{R}}(i),pos_C(j)}$, if $i \in R, j \in \bar{C}$ then it is $X^{1,2}_{pos_R(i),pos_{\bar{C}}(j)}$ and if $i \in \bar{R}, j \in \bar{C}$ then it is $X^{2,2}_{pos_{\bar{R}}(i),pos_{\bar{C}}(j)}$. This completes the proof. ◀

Phase 1. First, we restore the i-uniqueness of the columns indexed by the set C_1. Let R_{C_1} be the set of rows in R indexed by the same set of indices as C_1 in C. We right multiply M with another matrix $D^{(1)} \in \mathbb{Z}_p^{n\times n}$ such that $D^{(1)}_{R_{C_1},C_1}$ is the inverse of M_{R_1,C_1} and $D^{(1)}_{R-R_{C_1},C-C_1}$ is the identity matrix and all the other entries of $D^{(1)}$ are zero. Since the inverse of a $k \times k$ matrix can be computed in AC^0 [11], matrix $D^{(1)}$ can be obtained in AC^0 via Lemma 21.

Let $M^{(1)} = M \times D^{(1)}$, note that ${M^{(1)}}_{R_1,C_1}$ is an identity matrix. Since $M = A' \times B$, we have $M^{(1)} = M \times D^{(1)} = A' \times B \times D^{(1)}$. Note that since ${M^{(1)}}_{R_1,C_1}$ is an identity matrix, the vectors corresponding to the columns in C_1 in the matrix $B \times D^{(1)}$ can now easily be made i-unique. Since ${M^{(1)}}_{R_1,C_1}$ is an identity matrix, all columns in the matrix ${M^{(1)}}_{R_1,C-C_1}$ can be written as the linear combinations of columns of ${M^{(1)}}_{R_1,C_1}$. Let $\tilde{M}^{(1)}$ be a matrix defined as $\tilde{M}^{(1)} = M^{(1)} \times E^{(1)}$, where $E^{(1)} \in \mathbb{Z}_p^{n\times n}$ is constructed as follows. (i) $E^{(1)}_{R_{C_1},*}$ is same as $M^{(1)}_{R_1,*}$. (ii) $E^{(1)}_{R-R_{C_1},C_1}$ is the zero matrix. (iii) $E^{(1)}_{R-R_{C_1},C-C_1}$ is the negative identity matrix. Using Lemma 21, we can construct $E^{(1)}$ in AC^0. Note that $\tilde{M}^{(1)}_{R_1,C_1}$ is an identity matrix and $\tilde{M}^{(1)}_{R_1,C-C_1}$ is a zero matrix. Thus we can say that vectors corresponding to columns in C_1 in the matrix $B \times D^{(1)} \times E^{(1)}$ are i-unique for some $i \in R_1$.

Next, we perform a procedure similar to Phase 1 for those vectors which lost their pc's when we changed the matrix from A to A', i.e. those vectors w which were i-unique for some i, but $i \notin S'(w)$.

Phase 2. There can be at most k vectors which lost their pc's while changing the matrix from A to A'. Some of these vectors might get their pc's set in Phase 1. Let $\tilde{C}_2$ be the remaining set of vectors in B. Notice $C_1 \cap \tilde{C}_2$ is empty. To set the pc's of these vectors, we repeat the above procedure for the matrix $\tilde{M}^{(1)}_{*,\tilde{C}_2}$ as follows.

Find the column basis C_2 of $\tilde{M}^{(1)}_{*,\tilde{C}_2}$ in AC^0 recalling that $|\tilde{C}_2| \leq k$ and using Claim 19 on the transpose of $\tilde{M}^{(1)}_{*,\tilde{C}_2}$. By considering the transpose of $\tilde{M}^{(1)}_{*,C_2}$ and applying Claim 20 we can get a row basis R_2 of $\tilde{M}^{(1)}_{*,C_2}$. Notice that $R_1 \cap R_2$ is empty. We construct a matrix $D^{(2)} \in \mathbb{Z}_p^{n\times n}$ in AC^0 using Lemma 21 such that $D^{(2)}_{R_{C_2},C_2}$ contains the inverse of $\tilde{M}^{(1)}_{R_2,C_2}$, $D^{(2)}_{R-R_{C_2},C-C_2}$ is an identity matrix and the rest of the entries of $D^{(2)}$ are zero. Here R_{C_2} is the set of rows indexed by the same indices as in the set C_2 of columns. Let $\tilde{M}^{(2)} = \tilde{M}^{(1)} \times D^{(2)} \times E^{(2)}$, where $E^{(2)}$ is constructed in AC^0 (using Lemma 21) so that: (i) $\tilde{M}^{(2)}_{R_1\cup R_2,C_1\cup C_2}$ is the identity matrix. (ii) $\tilde{M}^{(2)}_{R_1,C-C_1}$ and $\tilde{M}^{(2)}_{R-R_1,\tilde{C}_2-C_2}$ are zero matrices. Finally, we have $B' = B \times D^{(1)} \times E^{(1)} \times D^{(2)} \times E^{(2)}$. Since each column of the newly constructed matrices contains at most $(k+1)$ non-zero entries, we can obtain B' in AC^0.

▷ Claim 22. B' is an A'-good basis.

Proof. First, we prove that the vectors which lost their pc's, either get new pc's or they are modified to be in the kernel of A'. Let $w \in B$ be a vector that lost its pc and it is not captured in both Phase 1 or Phase 2. Assume it is not captured in Phase 1, i.e. the vector $A'w$ does not belong to the column set indexed by C_1. Then it will be captured in Phase 2. If it is not captured in Phase 2 as well, then we can say that $A'w$ does not belong to the columns indexed by the set C_2. Therefore it can be written as a linear combination of the vectors in C_2. In Phase 2, we modify such vectors in a way that $A'w$ becomes a zero vector, i.e. w goes into the kernel of A'. Also, note that the vector which did not lose their pc's and are not captured in Phase 1 and Phase 2, do not lose their pc's in the procedure. ◁

We prove that we can maintain the number of pc's in B in AC^0 using the next claim. However, we need to set up some notation first. Let $P^{\text{old}}, P^{\text{new}}$ be respectively, the number of pc's before and after the phases. Let $V^{\text{new}}_{R_1}$ and $V^{\text{new}}_{R_2}$ denote the set of vectors that have their pc's in the rows R_1 and R_2, after the phases. Let $V^{\text{old}}_{R_0}$ denote the set of vectors that have their pc's in the rows R_0 before starting of Phase 1 and V_1 denotes the set of vectors which have their pc's in the rows $R - R_0$ before the Phase 1 and attain pc's in the rows R_1 after Phase 2. Note that all the cardinalities of all the sets of vectors mentioned above are $O(\frac{\log n}{\log\log n})$. Therefore, we can compute their cardinalities in AC^0.

▷ Claim 23. $P^{\text{new}} = P^{\text{old}} - |V^{\text{old}}_{R_0}| + |V^{\text{new}}_{R_1}| + |V^{\text{new}}_{R_2}| - |V_1|$.

Proof. First, we assume that all the vectors in the set $V^{\text{old}}_{R_0}$ lose their pc's after the phases therefore we subtract $|V^{\text{old}}_{R_0}|$ from P. But some of these vectors get their pc's in Phase 1 and Phase 2. Therefore, we add $|V^{\text{new}}_{R_1}|$ and $|V^{\text{new}}_{R_2}|$ back to the sum. Notice that $V^{\text{new}}_{R_1}$ may contain those vectors as well that had their pc's in the rows indexed by $R - R_0$ before Phase 1. That means these vectors had a pc before and after the two phases, but we added their number $|V^{\text{new}}_{R_1}|$. Therefore, we subtract the number of such vectors by subtracting $|V_1|$ from the total sum. ◁

This brings us to the proof of Lemma 15.

Proof of Lemma 15. The proof is complete from the above claims because the number of pc's is precisely the rank of the matrix as a consequence of Proposition 16. ◀

8 Conclusion

In this work, we prove two meta-theorems for distance and maximum matching, which provide the best known dynamic bounds in graphs where non-zero circulation weights can be computed in parallel. This includes important graph classes like planar, bounded genus, bounded treewidth graphs. We show how to non-trivially modify two known techniques and combine them with existing tools to yield the best known dynamic bounds for more general classes of graphs, and at the same time allow for bulk updates of larger cardinality. While for bipartite matching we are able to show a $\mathsf{DynFO}(\leq, +, \times)$ bound it would be interesting to achieve this also for maintaining distances, even in planar graphs under single edge changes.

References

1 Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Comput. Complex.*, 8(2):99–126, 1999.

2 David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within nc^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

3 Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Transactions on Computation Theory*, 1(1):1–17, 2009. doi:10.1145/1490270.1490274.

4 Samir Datta, Chetan Gupta, Rahul Jain, Anish Mukherjee, Vimal Raj Sharma, and Raghunath Tewari. Reachability and matching in single crossing minor free graphs. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

5 Samir Datta, William Hesse, and Raghav Kulkarni. Dynamic complexity of directed reachability and other problems. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 356–367. Springer, 2014.

6 Samir Datta, Raghav Kulkarni, Ashish Kumar, and Anish Mukherjee. Planar maximum matching: Towards a parallel algorithm. In *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, pages 21:1–21:13, 2018.

7 Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in DynFO. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2015.

8 Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in DynFO. *J. ACM*, 65(5):33:1–33:24, 2018.

9 Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory Comput. Syst.*, 47(3):737–757, 2010.

10 Samir Datta, Raghav Kulkarni, Raghunath Tewari, and N.V. Vinodchandran. Space complexity of perfect matching in bounded genus bipartite graphs. *Journal of Computer and System Sciences*, 78(3):765–779, 2012. In Commemoration of Amir Pnueli. doi:10.1016/j.jcss.2011.11.002.

11 Samir Datta, Pankaj Kumar, Anish Mukherjee, Anuj Tawari, Nils Vortmeier, and Thomas Zeume. Dynamic complexity of reachability: How many changes can we handle? In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 122:1–122:19, 2020.

12 Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. *Log. Methods Comput. Sci.*, 15(2), 2019.

13 Samir Datta, Anish Mukherjee, Nils Vortmeier, and Thomas Zeume. Reachability and distances under multiple changes. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 120:1–120:14, 2018.

14 Guozhu Dong, Jianwen Su, and Rodney W. Topor. Nonrecursive incremental evaluation of datalog queries. *Ann. Math. Artif. Intell.*, 14(2-4):187–223, 1995.

15 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in Quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 754–763, 2016.

16 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with O(1) worst case access time. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 165–169, 1982.

17 D. A. Harville. *Matrix Algebra From a Statistician's Perspective*, volume 8 of *Cambridge international series on parallel computation.* New York: Springer- Verlag, 2008.

18 Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 672–683, 2006.

19 Harold V Henderson and Shayle R Searle. On deriving the inverse of a sum of matrices. *Siam Review*, 23(1):53–60, 1981.

20 William Hesse. The dynamic complexity of transitive closure is in DynTC^0. *Theor. Comput. Sci.*, 296(3):473–485, 2003.

21 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

22 Thanh Minh Hoang. On the matching problem for special graph classes. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 139–150. IEEE Computer Society, 2010.

23 Neil Immerman. Expressibility and parallel complexity. *SIAM J. Comput.*, 18(3):625–638, 1989.

24 Vivek Anand T. Kallampally and Raghunath Tewari. Trading determinism for time in space bounded computations. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, pages 10:1–10:13, 2016.

25 László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

26 Anish Mukherjee. *Static and Dynamic Complexity of Reachability, Matching and Related Problems.* PhD thesis, CMI, 2019.

27 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 345–354, 1987.

28 Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997.

29 Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. *Inf. Comput.*, 215:1–7, 2012.

30 Wikipedia contributors. Matrix determinant lemma — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Matrix_determinant_lemma`, 2021. [Online; accessed 30-September-2021].