Fine-Grained Complexity Lower Bounds for Families of Dynamic Graphs

Monika Henzinger
Department of Computer Science, Universität Wien, Austria

Ami Paz
LISN, CNRS & Paris-Saclay University, France

A. R. Sricharan
Department of Computer Science, Universität Wien, Austria

Abstract

A dynamic graph algorithm is a data structure that answers queries about a property of the current graph while supporting graph modifications such as edge insertions and deletions. Prior work has shown strong conditional lower bounds for general dynamic graphs, yet graph families that arise in practice often exhibit structural properties that the existing lower bound constructions do not possess. We study three specific graph families that are ubiquitous, namely constant-degree graphs, power-law graphs, and expander graphs, and give the first conditional lower bounds for them. Our results show that even when restricting our attention to one of these graph classes, any algorithm for fundamental graph problems such as distance computation or approximation or maximum matching, cannot simultaneously achieve a sub-polynomial update time and query time. For example, we show that the same lower bounds as for general graphs hold for *maximum matching* and *(s,t)-distance* in constant-degree graphs, power-law graphs or expanders. Namely, in an m-edge graph, there exists no dynamic algorithms with both $O(m^{1/2-\epsilon})$ update time and $O(m^{1-\epsilon})$ query time, for any small $\epsilon > 0$. Note that for (s,t)-distance the trivial dynamic algorithm achieves an almost matching upper bound of constant update time and $O(m)$ query time. We prove similar bounds for the other graph families and for other fundamental problems such as densest subgraph detection and perfect matching.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases Dynamic graph algorithms, Expander graphs, Power-law graphs

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.65

Related Version *Full Version*: `https://arxiv.org/abs/2208.07572`

Funding *Monika Henzinger and A. R. Sricharan*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019564 "The Design of Modern Fully Dynamic Data Structures (MoDynStruct)" and from the Austrian Science Fund (FWF) project "Fast Algorithms for a Reactive Network Layer (ReactNet)", P 33775-N, with additional funding from the *netidee SCIENCE Stiftung*, 2020–2024

1 Introduction

A dynamic graph algorithm is a data structure that stores a graph and supports update operations, usually consisting of edge insertions and deletions, as well as query operations that ask about a specific property of the graph. The introduction of strong conditional lower bounds based on widely-believed complexity assumptions [1, 12] has had a fundamental influence on the field, pushing the design of new algorithms towards more specialized algorithms such as partially-dynamic or even offline-dynamic algorithms or towards approximate solutions. However, graphs arising in real-world applications often differ significantly from the very specifically crafted graphs for which the lower bound results are shown. Frequently, real-world

graphs have some special structure, such as having a power-law degree distribution, a constant degree, or being planar. Expanders, on the other side, have recently been used to design dynamic algorithms for *general* graphs. This naturally leads to the question of determining the complexity of dynamic graph algorithms for these graph classes, and this is exactly the question investigated in this paper.

While the complexity of dynamic graph algorithms for planar graphs has already been studied quite extensively [20, 14, 18, 16, 3, 2, 1, 15, 5], the question is still widely open for other families of graphs, including power-law graphs, constant-degree graphs, and expanders. Certain problems become easier for these graph classes: As an N-node[1] constant-degree graph has $O(N)$ edges, computing all-pairs shortest paths (APSP) takes only time $\tilde{O}(N^2)$, while the popular APSP conjecture postulates that for general graphs, there exists a small constant $c > 0$ such that any algorithm in the word RAM model with $O(\log N)$-bit words requires $N^{3-o(1)}$ expected time to compute APSP. Moreover, some problems become trivial in these graph classes, e.g., computing shortest paths with logarithmic additive error on expander graphs is trivial, due to their low diameter.

In this paper we will concentrate on graph problems that have real-world applications such as shortest-paths (which has applications in online navigation), matching (which has applications in reconfigurable datacenters), and densest subgraphs (which has applications in network analysis), yet we believe that our general approach can be applied to further graph problems. For these three problems, the known conditional lower bounds construct graphs that are far from being in the classes we consider: They have maximum degree $\Omega(N)$ and small cuts, and their degree distribution is unknown as it depends on the instance that is postulated to be hard.

Constant-degree graphs. Various dynamic graph problems that admit strong lower bounds in general graphs have very efficient algorithms on constant-degree graphs. Let Δ be the maximum degree in the graph. For *local* problems, where the solution at a node v can be computed by simply analyzing information stored at the neighbors of v such as maintaining a maximal matching, a maximal independent set, or a $(\Delta+1)$-vertex coloring, there exist simple dynamic algorithms with $O(\Delta)$ update time and constant query time. Additionally, for various problems that count or detect certain fixed subgraphs with c nodes (such as triangle counting for $c = 3$) there exists dynamic algorithms with $O(\Delta^{c-1})$ update time and constant query time, even though they have polynomial conditional lower bounds in general graphs (see Table 1). These efficient algorithms for local problems rule out the possibility of any non-trivial lower bound in the constant-degree setting.

Furthermore, even for the non-local problem of maintaining a maximum matching Gupta and Peng [11] designed a $(1+\delta)$-approximation algorithm for any small $\delta > 0$ that runs in $O\left(\min\left\{\frac{m}{|M(t)|}, |M(t)|\right\}\delta^{-2}\right)$ amortized time per update, where $M(t)$ denotes the maximum cardinality matching after the t-th update operation. As in a graph with maximum degree Δ it holds that $|M(t)| \geq m/(2\Delta)$, their algorithm achieves an amortized update time of $O(\Delta\,\delta^{-2})$, which is $O(\delta^{-2})$ in constant-degree graphs. This raises the question of how efficiently other non-local dynamic graph problems such exact maximum matching, shortest paths, and densest subgraph can be solved in dynamic constant-degree graphs and whether it is possible to show (conditional) lower bounds for them.

[1] To avoid confusion with the parameters n and M used in the online-matrix-vector multiplication conjecture, we use N to denote the number of vertices and m the number of edges in the dynamic graphs.

Table 1 Counting problems which admit polynomial conditional lower bounds on general graphs (amortized) and on Erdős-Rényi graphs (average case), but have algorithms with constant update and query times in constant-degree graphs. For the lower bounds above, there is no dynamic algorithm with pre-processing time $p(m, N)$, update time $u(m, N)$, and query time $q(m, N)$ unless the OMv conjecture is false. When $p(m, N)$ is unspecified, poly(N) pre-processing time is allowed.

Problems	Lower bounds					Upper bounds	
	General graphs [12]		Erdős-Rényi avg-case [13]			constant Δ (trivial)	
	$u(m,N)$	$q(m,N)$	$p(m,N)$	$u(m,N)$	$q(m,N)$	$u(m,N)$	$q(m,N)$
Triangle counting	$m^{1/2-\epsilon}$	$m^{1-\epsilon}$	$N^{3-\epsilon}$	$m^{1/2-\epsilon}$	$m^{1-\epsilon}$	Δ	1
C_4 subgraph counting						Δ^3	1
5-length (s,t)-path count			$N^{2-\epsilon}$	$N^{\omega-2-\epsilon}$	1	Δ^4	1

Expanders. Expander decompositions are increasingly becoming a central tool for designing dynamic graph algorithms with improved running time bounds for various graph problems such as connectivity, minimum spanning tree, shortest paths, conductance, edge-connectivity, maximum flows, and minimum cuts [17, 9, 7, 6]. One of the central subproblems that these algorithms have to handle is to solve a graph problem on a dynamically changing expander. To understand the limitations of this approach it is crucial to understand which problems can be solved efficiently on expanders, and which cannot. We present novel lower bounds for dynamic problems on expanders, more specifically on constant-degree expanders.

Note that these results also have an interesting connection to the average-case hardness of dynamic graph algorithms. Recently, lower bounds on the average-case hardness were shown for various subgraph counting problems in dynamic Erdős-Rényi graphs (see Table 1 for some of them) [13]. As random graphs are usually expanders, giving lower bounds for a problem on dynamic expanders gives an indication that this problem might also be hard in the average case and can motivate further work in this direction.

Power-law graphs. Graphs are called *power-law graphs* if the fraction of nodes with degree d is proportional to $1/d^c$ for some constant $c > 0$. Static and dynamic power-law graphs arise surprisingly often in real-world networks, such as the Internet, the world-wide web, and citation graphs, as well as in physics, linguistics, and economics. Even though the existence of large dynamic power-law graphs was already pointed out in 2004 [10], no efficient dynamic algorithms have been developed specifically for this class of graphs. This leads to the question of whether sub-polynomial time dynamic algorithms are even possible for power-law graphs or not. In fact, dynamic power-law graphs were not only never studied, they were not even defined – removing even a single edge from a power-law graph changes the degree distribution and thus violates the power-law distribution. Hence, we first present several definitions of *dynamic* power-law graphs, where some slackness in the degree-distribution is allowed. Then we prove lower bounds that hold for all of these dynamic power-law graph definitions.

1.1 Our Results

Throughout the paper we use the standard assumption that queries output one value, such as the size, length or weight of the solution. Note that this makes it only more challenging to prove lower bounds. All our results are conditioned on the popular OMv conjecture [12], defined in Section 2, but to simplify the terminology we usually drop the word "conditional". Our results are also summarized in Table 2.

1. *Main results.* We study the hardness of dynamic algorithms for (i) constant-degree graphs, (ii) expanders, and (iii) power-law graphs, for the following graph problems: Determining (a) the size of a maximum matching, (b) the length of the (s,t) shortest-path (i.e. (s,t)-*distance*), and (c) the density of the densest subgraph. Specifically, we show the following tradeoff between the update time u and the query time q in an m-edge graph for maximum matching and (s,t)-distance: There is no dynamic algorithm which achieves both $u = O(m^{1/2-\epsilon})$ and $q = O(m^{1-\epsilon})$ for any small $\epsilon > 0$. Note that these bounds match the bounds given for general graphs in [12] and that the lower bound for (s,t)-distance is almost tight as the simple algorithm that only records the edge change at update time and computes the solution from scratch at query time achieves $u = O(1)$ and $q = O(m)$. For densest subgraph we show that there is no dynamic algorithm which achieves both $u = O(N^{1/4-\epsilon})$ and $q = O(N^{1/2-\epsilon})$ for any small $\epsilon > 0$, which is weaker than the lower bound on general graphs (of $u = O(N^{1/2-\epsilon})$ and $q = O(N^{1-\epsilon})$).

   The only relevant prior work are conditional lower bounds for planar graphs [1], which have constant degree: In unweighted graphs they show for all-pairs-shortest paths a weaker tradeoff between update time u and query time q than we do, namely they prove $\max(u^2 \cdot q, u \cdot q^2) = \Omega(m^{1-o(1)})$. In *weighted* graphs they show for (s,t)-distance a tradeoff of $\max(u,q) = \Omega(m^{1/2-o(1)})$. Note that our result is stronger as it shows that in *unweighted* graphs no algorithm with $u = \Omega(m^{49/100})$ and $q = \Omega(m^{99/100})$ is possible.
2. *Degree–lower bound trade-off.* While the constant-degree lower bounds are equal to the lower bounds for general graphs in terms of m, they are naturally quadratically lower in terms of the number of nodes N. To understand the behaviour of the bounds also with respect to N, we extend our constant-degree lower bounds for maximum matching, perfect matching, and (s,t)-distance to graphs with maximum degree $O(N^t)$, for any $0 \leq t \leq 1$. We show the following result: There is no dynamic algorithm which achieves both $u = O(N^{(1+t)/2-\epsilon})$ and $q = O(N^{1+t-\epsilon})$ in a graph with maximum degree $O(N^t)$ for any $\epsilon > 0$. These results hold even in bipartite graphs. Note that for $t = 1$ these results match exactly the bounds for general graphs in [12], and for $t = 0$, they match the aforementioned results for constant-degree graphs.
3. *Approximation results.* In constant-degree graphs we extend the lower bound to the problem of approximating the (s,t)-distance within a factor of $3 - \delta$, for any small constant δ. This naturally extends the $(3-\delta)$-approximation lower bounds on general graphs to the constant-degree case. In planar graphs, the (s,t)-distance lower bound holds only for exact answers.

   Note that a similar extension to approximation algorithms is not possible for maximum cardinality matching and for densest subgraph: (a) For maximum matching, for any small $\delta > 0$ the above-mentioned $(1+\delta)$-approximation algorithm [11] achieves an amortized update time of $O(\delta^{-2})$, which is constant for a constant δ, thereby precluding any non-trivial lower bounds for approximate maximum matching in the constant-degree setting. Stated differently, *our work shows an interesting dichotomy for dynamic matching matching in constant-degree graphs*: For the exact setting there is no dynamic algorithm which achieves both $u = O(m^{1/2-\epsilon})$ and $q = O(m^{1-\epsilon})$ for any small $\epsilon > 0$, while a $(1+\delta)$-approximation can be achieved in constant time, for any small $\delta > 0$. (b) *The same dichotomy arises for densest subgraph*: For any small $\delta > 0$ there exists a $(1-\delta)$-approximation algorithm with polylogarithmic time per operation [19], while we show a polynomial lower bound for the exact value.
4. *Partially dynamic algorithms.* We extend the constant-degree reductions for maximum matching and (s,t)-distance also to the insertions-only and the deletions-only setting, achieving the same lower bound as in the fully dynamic setting.

Table 2 Our results for graphs on N nodes with m edges. For every u and q stated above, there is no algorithm for the corresponding problem with amortized $O(u(m,N))$ update time and $O(q(m,N))$ query time simultaneously unless the OMv conjecture is false. The first three rows hold also for perfect matching. All the lower bounds in the table except for densest subgraph match the general OMv lower bounds.

Problem	Class	$u(m,N)$	$q(m,N)$
Maximum Matching	$\Delta \leq 3$	$m^{1/2-\epsilon}$	$m^{1-\epsilon}$
	constant degree & expansion		
	power-law graphs		
	$\Delta \leq 3$, partially dynamic		
	$\Delta \leq N^t$	$N^{(1+t)/2-\epsilon}$	$N^{1+t-\epsilon}$
(s,t)-distance	$\Delta \leq 3$	$m^{1/2-\epsilon}$	$m^{1-\epsilon}$
	$(3-\delta)$-approx, $\Delta \leq 3$		
	constant degree & expansion		
	power-law graphs		
	$\Delta \leq 3$, partially dynamic		
	$\Delta \leq N^t$	$N^{(1+t)/2-\epsilon}$	$N^{1+t-\epsilon}$
Densest Subgraph	$\Delta \leq 5$	$N^{1/4-\epsilon}$	$N^{1/2-\epsilon}$
	constant degree & expansion		
	power-law graphs		

5. *Perfect matching.* A special case of maximum cardinality matching is determining whether a perfect matching exists in a bipartite graph. For constant-degree graphs and expander graphs we show the following lower bound: There is no dynamic algorithm which achieves both $u = O(m^{1/2-\epsilon})$ and $q = O(m^{1-\epsilon})$ for any small $\epsilon > 0$. This can also be extended to the varying-degree setting.

To summarize, our paper opens up the research field of dynamic graph algorithms for more specific, practical graph classes, in contrast with previous work that concentrated on general or planar graphs. We believe that our techniques can be extended to further classes of dynamic graphs or even in other domains of theoretical computer science, such as distributed graph algorithms or streaming algorithms. One further interesting implication of our work is presenting the limitations of dynamic graph algorithms on expanders, thus complementing recent algorithmic results that use expander decompositions in dynamic graphs.

1.2 Our Techniques

We prove lower bounds by reductions from the online matrix vector (OMv) conjecture [12]. In these reductions, the input of an online problem, which is an $n \times n$ matrix M and a sequence of n pairs (u,v) of n-vectors, is translated into a dynamic graph. The reduction is built so that there exists a pair (u,v) satisfying $uMv = 1$ if and only if the dynamic graph has some desired property at some point of time. While we follow the general framework of OMv lower bounds, the details are delicate, as the dynamic graphs we construct should fall into specific graph classes at all times, while still maintaining the graph property under consideration. We give a high-level overview of our reductions below.

One way to turn known OMv-to-dynamic graphs reductions into reductions that produce bounded-degree graphs is by replacing high-degree nodes by bounded-degree trees. This technique has a rather clear and straightforward effect on the distances in the graph, so it is applicable when considering distance-related problems. This, however, is far from being

the case when considering other problems, such as maximum matchings. Here, replacing a high-degree node with a gadget could adversely affect the desired matching size, since the gadget might create several augmenting paths that would not have existed when it was a single high-degree node. To overcome this, we limit the possible maximum matching sizes, by designing a reduction graph with bounded-degree gadgets composed of paths, where the maximum matching is always either a perfect matching, or a near-perfect matching, i.e., the matching size is either $N/2$ or $N/2 - 1$. This reduction thus involves a large matching and a small gap between the $uMv = 0$ and $uMv = 1$ cases, and hence cannot be extended to achieve a lower bound for the approximation of the maximum matching size. While this might seem as a limitation of our construction, recall that this is actually not the case: As described above, for any small $\delta > 0$ there is a constant time $(1 + \delta)$-approximation dynamic algorithm for the problem, and, thus, such a lower bound cannot exist.

An even more delicate reduction we present is for proving a lower bound on the densest-subgraph problem. A straightforward reduction would change $O(n)$ graph-edges for every bit of the input, which will allow us to make sure that the density of the densest subgraph changes by a significant amount when $uMv = 0$ versus when $uMv = 1$. However, this would involve $O(n^2)$ updates for each (u, v) input pair, and the reduction would fail to yield any non-trivial lower bound. Thus, we are forced to change very few edges for each input bit, which renders an almost negligible effect on the density, making it difficult to control the exact density of the densest subgraph. Our reduction balances these two factors, using a construction where each gadget is a sufficiently dense regular graph, while having each bit of the input translate into the existence or nonexistence of merely two edges inside specific gadgets. As in the case of matchings, our lower bounds cannot be extended to approximations, as for any $\delta > 0$ there exists a fast algorithm with polylogarithmic update time for computing $(1 - \delta)$-approximations to the densest subgraph.

We then extend these reductions from bounded-degree graphs to constant-degree *constant-expansion* graphs. First, the standard lower bound reductions contain sparse cuts if the inputs M, u or v are sparse, making a standard reduction graph far from being an expander. Thus, we have to augment the graph with many more edges to make sure that it has no sparse cuts regardless of M, u and v. We do this augmentation "inside a layer" to prevent the additional edges from creating undesired short paths between s and t, or spurious augmenting paths in the case of matchings. Sparse cuts also exist in parts of the graph that do not depend on M, u or v, and to handle these, we add edges of a constant-degree expander between a well-chosen set of nodes, thus guaranteeing the expansion without changing the required graph property. Finally, in the case of distance-related problems, we note that expander graphs can have at most logarithmic diameter, but the substitution of nodes by trees described above increases the diameter to be at least logarithmic, leaving only a very small slack for our construction.

When studying densest subgraphs on expanders, adding edges in order to avoid sparse cuts might change the location and structure of the densest subgraph in an undesired way. In order to guarantee the expansion in this case, we add a copy of all the graph nodes, build a constant-degree expander on the copies of the nodes, and then connect each node to its copy by a matching.

In dynamic power-law graphs where the node degrees may depend on the inputs u, M, v and change over time, we have to guarantee that the degree changes incurred by the processing of different inputs do not cause a violation of the power-law distribution of degrees. As before, all the changes must also be done without changing the graph property under consideration, and without performing too many update operations. We address this problem by inserting or deleting edges in an online fashion in other parts of the reduction graph, to compensate for the changes incurred by processing the input vector pairs.

Organisation. Section 2 has notation and definitions. Section 3 presents the dynamic maximum matching lower bounds. The lower bounds for other problems, namely densest subgraph detection and (s,t)-distance, are briefly discussed in Section 4, and so is the partially dynamic setting. The full lower bounds, the results for partially dynamic graphs, and the missing proofs all appear in the full version of the paper.

2 Preliminaries

Throughout the paper, we consider vectors and matrices that are boolean, and so a vector-matrix-vector multiplication outputs a single bit. Henzinger *et al.* [12] define the *Online Matrix Vector* (OMv) and the *Online Vector Matrix Vector* (OuMv) multiplication problems.

▶ **Definition 1** (Online Matrix Vector Multiplication). *Let M be a boolean $n \times n$ matrix. Preprocessing the matrix is allowed. Then, n vectors $v^1, v^2, \ldots, v^n$ arrive one at a time, and the task is to output the product Mv^i before the next vector is revealed.*

▶ **Definition 2** (Online Vector Matrix Vector Multiplication). *Let M be a boolean $n \times n$ matrix. Preprocessing the matrix is allowed. Then, n vector pairs $(u^1, v^1), (u^2, v^2), \ldots, (u^n, v^n)$ arrive one at a time, and the task is to output the bit $u^i M v^i$ before the next vector pair is revealed.*

In their paper, they show that the OuMv problem can be reduced to the OMv problem, and conjecture that there is no truly subcubic time algorithm for OMv.

▶ **Conjecture 3** (OMv). *There is no algorithm for the* OMv *(and hence the* OuMv*) problem running in time $O(n^{3-\epsilon})$ for any constant $\epsilon > 0$.*

We work with the OuMv problem for all the reductions in our paper. We denote the length of our input vectors u^i, v^i by n, and thus the matrix M is of dimension $n \times n$. We use upper indices to indicate the vector's location in the stream, but usually focus on one pair (u, v) omitting these indices. We use lower indices for a location in the vector or matrix, e.g., u_i and M_{ij}. We use N to denote the number of nodes in our reduction graph.

▶ **Definition 4** (Expansion). *The expansion parameter of a graph $G = (V, E)$ is defined as*

$$h = \min \left\{ \frac{|E(S, \overline{S})|}{|S|} \;\middle|\; \emptyset \neq S \subseteq V,\ |S| \leq |V|/2 \right\}$$

where $E(S, \overline{S})$ is the number of edges from S to $V \setminus S$. We call a graph with expansion h a *h-expander*. Works on dynamic algorithms use a different definition of expansion parameter h', called *volume expansion*. However, when considering constant-degree graphs with constant expansion (as we do in this paper), both parameters are within a Δ factor of each other, so we only consider the expansion parameter h in our proofs.

We study *power-law* graphs as introduced in [4], and only consider the setting where $\beta > 2$. In the following definition, if the number of nodes N in the graph is fixed, then we get that α is roughly $\ln N$.

▶ **Definition 5** (Power Law). *A graph is said to follow an (α, β)-power law distribution if the number N_d of nodes with degree d is inversely proportional to d^β for some constant $\beta > 0$. That is,*

$$N_d = \left\lfloor \frac{e^\alpha}{d^\beta} \right\rfloor \approx \left\lfloor \frac{1}{\zeta(\beta)} \cdot \frac{N}{d^\beta} \right\rfloor,$$

where $\zeta(\beta) = \sum_{i=1}^{\infty} 1/i^\beta$ is the Riemann Zeta function.

Since dynamic graphs allow edge insertions and deletions, it is impossible to maintain an exact degree distribution at all times. Hence, we introduce the notion of approximate power-law distributions to afford some slack for dynamic changes. One natural relaxation is to allow β to vary within an interval.

▶ **Definition 6** (β-Varying Power Law). *A graph is said to follow an $(\alpha, \beta_1, \beta_2)$-varying power law distribution if the number N_d of nodes with degree d satisfies*

$$\min\left\{\left\lfloor\frac{1}{\zeta(\beta_1)}\cdot\frac{N}{d^{\beta_1}}\right\rfloor,\left\lfloor\frac{1}{\zeta(\beta_2)}\cdot\frac{N}{d^{\beta_2}}\right\rfloor\right\}\leq N_d\leq\max\left\{\left\lfloor\frac{1}{\zeta(\beta_1)}\cdot\frac{N}{d^{\beta_1}}\right\rfloor,\left\lfloor\frac{1}{\zeta(\beta_2)}\cdot\frac{N}{d^{\beta_2}}\right\rfloor\right\},$$

This relaxation of an exact power law, while being natural, is a global relaxation rather than a local one. Thus we also define two locally approximate definitions below that allow similar slack for all degrees.

▶ **Definition 7** (Additively Approximate Power Law). *A graph is said to follow an (α, β, c)-additively approximate power law distribution if the number N_d of nodes of degree d for a realisable degree d satisfies*

$$\left\lfloor\frac{1}{\zeta(\beta)}\cdot\frac{N}{d^{\beta}}\right\rfloor-c\leq N_d\leq\left\lfloor\frac{1}{\zeta(\beta)}\cdot\frac{N}{d^{\beta}}\right\rfloor+c$$

where we say that d is a realisable degree if there is a node of degree d in an (α, β)-power law graph.

▶ **Definition 8** (Multiplicatively Approximate Power Law). *A graph is said to follow an $(\alpha, \beta, \epsilon)$-multiplicatively approximate power law distribution if the number N_d of nodes of degree d satisfies*

$$\frac{1}{(1+\epsilon)}\cdot\left\lfloor\frac{1}{\zeta(\beta)}\cdot\frac{N}{d^{\beta}}\right\rfloor\leq N_d\leq(1+\epsilon)\cdot\left\lfloor\frac{1}{\zeta(\beta)}\cdot\frac{N}{d^{\beta}}\right\rfloor$$

Our lower bounds contain at most four nodes that are one degree away from an exact power-law distribution, and thus hold in all the models discussed above with any reasonable parameter regime.

3 Lower Bounds for Dynamic Maximum Matching

The previous matching lower bounds on general graphs [12, 8] use reduction graphs that contain nodes with degree $\Omega(N)$. In this section, we construct a constant-degree reduction graph with constant expansion.

Reduction gadgets

Our gadget construction starts by replacing each node of a dense reduction by a path; we refer to each path as a "subgadget".

Connecting every node of this new subgadget with nodes outside the subgadget might create unwanted matchings of larger sizes, so instead we carefully choose a subset of the path nodes to connect outside the subgadget. Figure 1 shows odd and even paths ("odd" and "even" describe the number of nodes) with a "canonical" matching for each of them marked in red. Next, we detail the connections outside the subgadgets.

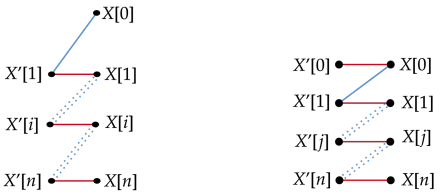

Figure 1 Odd and even sized paths used in the maximum matching lower bounds. The canonical matchings are marked in red.

Consider an odd path on $2n+1$ vertices, and a bipartition of the vertices into (X', X) with $|X'| \leq |X|$. Indexing the vertices as $X[0]$ and $X'[i], X[i]$ for $1 \leq i \leq n$, our canonical matching matches $X[i]$ with $X'[i]$, and leaves $X[0]$ unmatched. We connect only the vertices $\{X[i] \mid 1 \leq i \leq n\}$ outside the subgadget, while vertices in X' and $X[0]$ only have edges inside the subgadget. For an even path on $2n+2$ vertices indexed as above, our canonical matching is perfect, and matches $X[i]$ to $X'[i]$. Only vertex $X'[0]$ and all the vertices in X are connected outside the subgadget, and all vertices $X'[i]$, $1 \leq i \leq n$, only have edges within the subgadget.

While this suffices for sparsification, we need additional constructions in order to guarantee constant expansion. In particular, it turns out that adding edges inside a subgadget does not suffice for constant expansion, and we are forced to add edges between subgadgets. Our construction adds edges on the same side of the bipartition across subgadgets, and we show that if the newly added edges take part in any augmenting path, then there also exists an augmenting path in the subgraph devoid of any newly inserted edge.

The reduction graph consists of a left subgraph (L) and a right subgraph (R), connected together by edges corresponding to the matrix M. Note that for constant expansion, we need the number of edges of M to be a constant fraction of the sizes of L and R. While it would be possible to construct a reduction graph with $|L|$ and $|R|$ that depend on the input matrix M, we instead choose to augment the input matrix and vectors as it simplifies notation. We thus augment the input beforehand to ensure that there are $\Omega(n^2)$ edges crossing from L to R. To this end, we work with the vectors $\hat{u} = (u\ 0)$ and $\hat{v} = (v\ 0)$ of dimension $2n$, and the matrix $\hat{M} = \left(\begin{smallmatrix} M & 1 \\ 1 & 1 \end{smallmatrix}\right)$ of dimension $2n \times 2n$. It is easy to see that $uMv = 1 \iff \hat{u}\hat{M}\hat{v} = 1$.

▶ **Definition 9** (Reinforced gadget). *A* reinforced gadget *with x subgadgets of size y consists of x subgraphs, each of which is a path on y nodes. The nodes are bipartitioned into sets (X', X) with the larger side of the partition labelled as X in each subgadget. Thus $|X'| \leq |X|$. It is then augmented with the following edge-set: Consider a degree-d expander graph on $x \cdot \lceil \frac{y}{2} \rceil$ nodes, choose an arbitrary bijection between the expander nodes and X, and add the expander edges to these nodes accordingly. The resulting graph is the reinforced gadget.*

Note that reinforced gadgets are not bipartite. Thus, while the constant-degree lower bounds hold for bipartite matching, the expander result is for maximum matching on general graphs. In what follows, we drop the hats from $\hat{u}, \hat{M}$ and $\hat{v}$ for simplicity, but continue our analysis with their dimensions as $2n, 2n \times 2n, 2n$ respectively.

Reduction Graph

We use the following reduction graph, composed of two odd-sized reinforced gadgets and two even-sized reinforced gadgets.

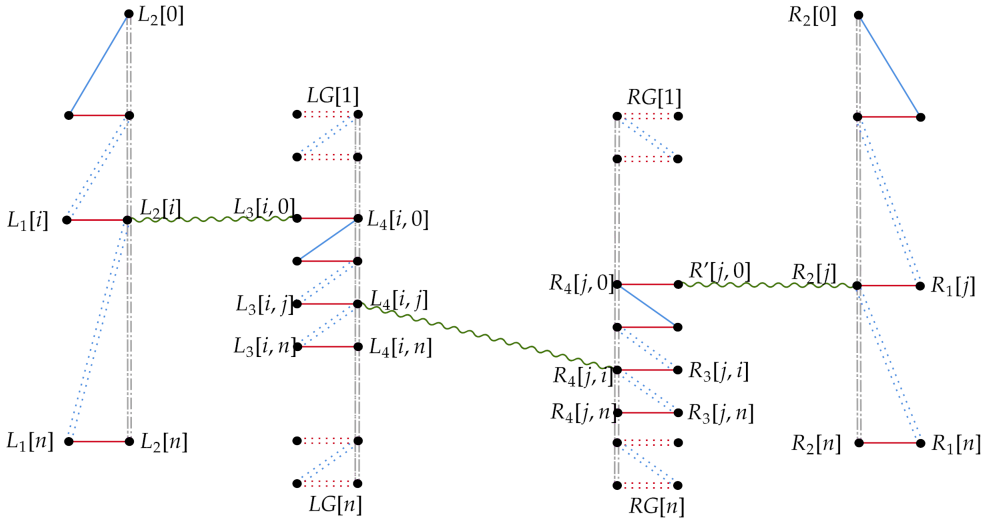

Figure 2 The expander reduction graph for maximum matching. The red lines denote the canonical matching, the blue lines denote the paths in each subgadget, the grey lines denote the expander edges, and the green lines denote the input-dependent edges.

- A reinforced gadget with one subgadget of size $4n+1$, on a set $L_1 \cup L_2$. The nodes are labelled $L_1[i]$ for $1 \le i \le 2n$, and $L_2[i]$ for $0 \le i \le 2n$. The path is from $L_2[0]$ to $L_2[2n]$, and the expander is on L_2.
- A reinforced gadget with $2n$ subgadgets of size $4n+2$, on a set $L_3 \cup L_4$. The subgadgets are labelled $LG[i]$ for $1 \le i \le 2n$, and the nodes of subgadget $LG[i]$ are labelled $L_3[i,j]$ or $L_4[i,j]$ for $0 \le j \le 2n$ depending on whether the node is in L_3 or L_4. The path in each subgadget goes from $L_3[i,0]$ to $L_4[i,2n]$, and the expander is on L_4.
- A copy of the above structure, with node sets marked R_i instead of L_i, respectively.
- For the matrix M, add the edge $(L_4[i,j], R_4[j,i])$ if $M_{ij} = 1$.
- Given an input vector u, for each $i \in [2n]$, add the edge $(L_2[i], L_3[i,0])$ if $u_i = 1$, and $(L_2[i], L_4[i,0])$ otherwise.
- Given an input vector v, for each $j \in [2n]$, add the edge $(R_2[j], R_3[j,0])$ if $v_j = 1$, and add the edge $(R_2[j], R_4[j,0])$ otherwise.

The total number of nodes in the reduction graph is $N = 16n^2 + 16n + 2 = \Theta(n^2)$.

Matchings in the Graph

We start by defining a base matching B on the graph, which is made up of the canonical matchings on each of the gadgets. On the left side, B matches $L_3[i,j]$ to $L_4[i,j]$, and $L_1[i]$ to $L_2[i]$ for all i,j. The matching on the right side is similar. Note that this matching always exists regardless of the input, and only $L_2[0]$ and $R_2[0]$ are unmatched in the entire graph. Thus $|B| = \frac{N}{2} - 1$. We claim that this graph has a perfect matching if and only if $uMv = 1$. Let C denote the maximum cardinality matching.

▶ **Lemma 10.** *If $uMv = 1$, then $|C| = \frac{N}{2}$, and otherwise $|C| = \frac{N}{2} - 1$.*

Proof. Since B is always a matching of size $\frac{N}{2} - 1$ regardless of the input, the claim is equivalent to showing that $uMv = 1$ if and only if there is an augmenting path with respect to the matching B.

($\implies$) Suppose that $uMv = 1$, with $u_i = M_{ij} = v_j = 1$. Consider the path P composed of the following subpaths, of which all except P_4 start with an unmatched edge and end with a matched edge, while P_4 both starts and ends with an unmatched edge.

- $P_1 = L_2[0], L_1[1], L_2[1], \ldots, L_2[i]$
- $P_2 = L_2[i], L_3[i, 0], L_4[i, 0], \ldots, L_4[i, j]$
- $P_3 = L_4[i, j], R_4[j, i], R_3[j, i], \ldots, R_3[j, 0]$
- $P_4 = R_3[j, 0], R_2[j], R_1[j], \ldots, R_2[0]$

Thus, P is an augmenting path to the base matching B, which gives us that the maximum matching C has to have size $> \frac{N}{2} - 1$, implying that the maximum matching C is a perfect matching.

($\impliedby$) Suppose now that there exists an augmenting path P to the base matching B that starts at $s = L_2[0]$ and ends at $t = R_2[0]$.

- Since $(\cup_i L_i, \cup_j R_j)$ is an (s, t)-cut, there is at least one crossing edge, say $(L_4[i, j], R_4[j, i])$, in P. Thus $M_{ij} = 1$.
- Since P leaves the subgadget $LG[i]$ using $(L_4[i, j], R_4[j, i])$, it should have entered the subgadget at some previous instance. Since $(L_4[i, j], R_4[j, i])$ is an unmatched edge and all the matching edges in $LG[i]$ are within the subgadget, P should have entered the subgadget using an unmatched edge. As all the matching edges in $LG[i]$ are between L_3 and L_4, P cannot both enter and exit the subgadget through L_4. Thus P enters $LG[i]$ through L_3. However, the only possible unmatched edge from L_3 leaving the subgadget is the edge $(L_3[i, 0], L_2[i])$. Thus P uses the edge $(L_3[i, 0], L_2[i])$ to enter the subgadget $LG[i]$, and so $u_i = 1$.
- The path P now enters the subgadget $RG[j]$ through the unmatched edge $(L_4[i, j], R_4[j, i])$. As before, all the matched edges in $RG[j]$ are between R_4 and R_3, and so P has to exit the subgadget using an unmatched edge from R_3. However, the only possible unmatched edge from R_3 leaving the subgadget is the edge $(R_3[j, 0], R_2[j])$. Thus the edge $(R_3[j, 0], R_2[j])$ is used by P, giving us that $v_j = 1$.

This gives us that $uMv = 1$ as required. ◀

Complexity of the Reduction

On the arrival of a new vector pair, we first add all the edges corresponding to the new input (if they do not already exist), and then remove the previous vector pair's edges, as opposed to the usual convention of first deleting the previous edges and inserting the new ones. This ensures that the graph remains an expander at all steps. The proof of constant expansion is involved, and we defer it to the full version. Since number of edges in a constant-degree graph is $O(N)$, we get the following theorem for expanders.

▶ **Theorem 11.** *For any constant $\epsilon > 0$, there is no dynamic algorithm maintaining a maximum matching or determining the existence of a perfect matching, on all N-node graphs with constant degree and constant expansion, with amortized $O(N^{1/2-\epsilon})$ update time and $O(N^{1-\epsilon})$ query time, unless the* OMv *conjecture is false.*

Proof. Consider the reduction graph above. It consists of $N = 16n^2 + 16n + 2 = \Theta(n^2)$ nodes. Every time we get a new (u, v) input vector pair, we update $L_2 \times L_3$ and $R_2 \times R_3$ as detailed above. This takes $O(n)$ updates in total. After that, we query once for the size of the maximum matching in this new graph, and return 1 if and only if $|C| = \frac{N}{2}$.

Thus for each pair of input vectors, we perform $O(n)$ updates and $O(1)$ query. In total, checking n vector pairs takes us $O(n^2)$ updates and $O(n)$ query. If there were an algorithm for maximum matching on constant-degree graphs with update time $O(N^{1/2-\epsilon})$ (i.e., $O(n^{1-2\epsilon})$) and query time $O(N^{1-\epsilon})$ (i.e., $O(n^{2-2\epsilon})$), then we can decide if $uMv = 1$ for all n pairs in $O(n^{3-2\epsilon})$ time, contradicting the OMv conjecture. ◀

4 Other Lower Bounds

Dynamic Maximum Matching. All our lower bounds for the dynamic maximum matching problem are summarized in Theorem 12. Note that $m = O(N)$ for the first three graph classes below.

▶ **Theorem 12.** *For any constant $\epsilon > 0$, there is no dynamic algorithm for maintaining the size of the maximum matching on the following graph classes, with the amortized update time $u(N) = O(N^{1/2-\epsilon})$ and amortized query time $q(N) = O(N^{1-\epsilon})$,*

1. $\Delta \leq 3$,
2. *constant degree, constant expansion,*
3. *power-law graph,*
4. $\Delta \leq O(N^t)$ *with* $u(N) = O(N^{(1+t)/2-\epsilon})$, *and* $q(N) = O(N^{1+t-\epsilon})$,

unless the OMv *conjecture is false.*

Dynamic Densest Subgraph. Our lower bounds for the dynamic densest subgraph problem are summarized in Theorem 13.

▶ **Theorem 13.** *For any constant $\epsilon > 0$, there is no dynamic algorithm for maintaining the density of the densest subgraph on the following graph classes, with the amortized update time $u(N) = O(N^{1/4-\epsilon})$ and amortized query time $q(N) = O(N^{1/2-\epsilon})$,*

1. $\Delta \leq 7$,
2. *constant degree, constant expansion,*
3. *power-law graph,*

unless the OMv *conjecture is false.*

Dynamic (s,t)-distance. Our lower bounds for the dynamic (s,t)-shortest paths problem are summarized in Theorem 14, and extend naturally to the SSSP and the APSP problems. Note that $m = O(N)$ for the first four classes below.

▶ **Theorem 14.** *For any constant $\epsilon > 0$, there is no dynamic algorithm for maintaining (s,t)-distance, SSSP, or APSP on the following graph classes, even for bipartite graphs, with the amortized update time $u(N) = O(N^{1/2-\epsilon})$ and amortized query time $q(N) = O(N^{1-\epsilon})$ (except for the last class), unless the* OMv *conjecture is false.*

1. $\Delta \leq 3$,
2. $(3-\delta)$*-approximation,* $\Delta \leq 3$,
3. *constant degree, constant expansion,*
4. *power law graph,*
5. $\Delta \leq O(N^t)$ *with* $u(N) = O(N^{(1+t)/2-\epsilon})$, *and* $q(N) = O(N^{1+t-\epsilon})$,

Partially Dynamic Algorithms. We summarize our lower bounds for partially dynamic algorithms in Theorem 15.

▶ **Theorem 15.** *For any constant $\epsilon > 0$, there is no algorithm in either the insertions-only or deletions-only setting maintaining (s,t)-distance or maximum matching, on all N-node graphs with maximum degree $\Delta \leq 3$, with amortized $O(N^{1/2-\epsilon})$ update time and $O(N^{1-\epsilon})$ query time, unless the* OMv *conjecture is false.*

References

1. Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *FOCS*, pages 477–486. IEEE Computer Society, 2016.
2. Ittai Abraham, Shiri Chechik, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. On dynamic approximate shortest paths for planar graphs with worst-case costs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 740–753. SIAM, 2016. doi:10.1137/1.9781611974331.ch53.
3. Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1199–1218, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214084.
4. William Aiello, Fan Chung Graham, and Linyuan Lu. A random graph model for power law graphs. *Exp. Math.*, 10(1):53–66, 2001. doi:10.1080/10586458.2001.10504428.
5. Panagiotis Charalampopoulos and Adam Karczmarz. Single-source shortest paths and strong connectivity in dynamic planar graphs. In *ESA*, volume 173 of *LIPIcs*, pages 31:1–31:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
6. Julia Chuzhoy. Decremental all-pairs shortest paths in deterministic near-linear time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 626–639. ACM, 2021. doi:10.1145/3406325.3451025.
7. Julia Chuzhoy and Thatchaphol Saranurak. Deterministic algorithms for decremental shortest paths via layered core decomposition. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2478–2496. SIAM, 2021. doi:10.1137/1.9781611976465.147.
8. Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. *CoRR*, abs/1602.06705, 2016. arXiv:1602.06705.
9. Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2212–2228. SIAM, 2021. doi:10.1137/1.9781611976465.132.
10. Fan Chung Graham. Large dynamic graphs: What can researchers learn from them? *SIAM News*, 37(3), 2004.
11. Manoj Gupta and Richard Peng. Fully dynamic (1+ ϵ)-approximate matchings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 548–557, 2013. doi:10.1109/FOCS.2013.65.
12. Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, pages 21–30. ACM, 2015.
13. Monika Henzinger, Andrea Lincoln, and Barna Saha. The complexity of average-case dynamic subgraph counting. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–498. SIAM, 2022.
14. Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997. doi:10.1006/jcss.1997.1493.

15 Giuseppe F. Italiano, Adam Karczmarz, Jakub Lacki, and Piotr Sankowski. Decremental single-source reachability in planar digraphs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1108–1121. ACM, 2017. doi:10.1145/3055399.3055480.

16 Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 313–322. ACM, 2011. doi:10.1145/1993636.1993679.

17 Wenyu Jin and Xiaorui Sun. Fully dynamic c-edge connectivity in subpolynomial time. *CoRR*, abs/2004.07650, 2020. arXiv:2004.07650.

18 P. N. Klein and S. Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, November 1998. doi:10.1007/PL00009223.

19 Saurabh Sawlani and Junxing Wang. Near-optimal fully dynamic densest subgraph. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 181–193, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384327.

20 Sairam Subramanian. A fully dynamic data structure for reachability in planar digraphs. In Thomas Lengauer, editor, *Algorithms - ESA '93, First Annual European Symposium, Bad Honnef, Germany, September 30 - October 2, 1993, Proceedings*, volume 726 of *Lecture Notes in Computer Science*, pages 372–383. Springer, 1993. doi:10.1007/3-540-57273-2_72.