Triangle Detection in H-Free Graphs

Amir Abboud
Weizmann Institute of Science, Rehovot, Israel

Ron Safier
Weizmann Institute of Science, Rehovot, Israel

Nathan Wallheimer
Weizmann Institute of Science, Rehovot, Israel

Abstract

We initiate the study of combinatorial algorithms for Triangle Detection in H-free graphs. The goal is to decide if a graph that forbids a fixed pattern H as a subgraph contains a triangle, using only "combinatorial" methods that notably exclude fast matrix multiplication. Our work aims to classify which patterns admit a subcubic speedup, working towards a dichotomy theorem.

On the lower bound side, we show that if H is not 3-colorable or contains more than one triangle, the complexity of the problem remains unchanged, and no combinatorial speedup is likely possible.

On the upper bound side, we develop an embedding approach that results in a strongly subcubic, combinatorial algorithm for a rich class of "embeddable" patterns. Specifically, for an embeddable pattern of size k, our algorithm runs in $\widetilde{O}(n^{3-\frac{1}{2^k-3}})$ time, where $\widetilde{O}(\cdot)$ hides poly-logarithmic factors. This algorithm also extends to listing all the triangles within the same time bound. We supplement this main result with two generalizations:

- A generalization to patterns that are embeddable up to a single obstacle that arises from a triangle in the pattern. This completes our classification for small patterns, yielding a dichotomy theorem for all patterns of size up to eight.
- An H-sensitive algorithm for embeddable patterns, which runs faster when the number of copies of H is significantly smaller than the maximum possible $\Omega(n^k)$.

Finally, we focus on the special case of odd cycles. We present specialized Triangle Detection algorithms that are very efficient:

- A combinatorial algorithm for C_{2k+1}-free graphs that runs in $\widetilde{O}(m + n^{1+2/k})$ time for every $k \geq 2$, where m is the number of edges in the graph.
- A combinatorial C_5-sensitive algorithm that runs in $\widetilde{O}(n^2 + n^{4/3}t^{1/3})$ time, where t is the number of 5-cycles in the graph.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases fine-grained complexity, triangle detection, H-free graphs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.1

Related Version *Full Version*: `https://arxiv.org/abs/2511.17224`

Supplementary Material *Software*: `https://github.com/RonSafier/Triangle-Detection-in-H-free-Graphs`, archived at `swh:1:dir:53adc96aa72369c0b705cde725ec29a28db203ef`

Funding This work is part of the project CONJEXITY that has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101078482). Supported by a research grant from the Center for New Scientists at the Weizmann Institute of Science.
Amir Abboud: Parts of this work were done while visiting INSAIT, Sofia University "St. Kliment Ohridski", Rutgers University, and the Courant Institute of Mathematical Sciences at New York University.

Acknowledgements We would like to thank Elad Tzalik for useful discussions.

1 Introduction

In the Triangle Detection problem, a graph G is given as input, and the goal is to decide whether G contains a triangle, i.e., three vertices such that an edge connects every pair of them. The trivial algorithm that checks every triplet of vertices runs in $O(n^3)$ time, and the only other known algorithm for the problem achieving strongly subcubic, i.e., $O(n^{3-\varepsilon})$ time for some $\varepsilon > 0$, is the $O(n^\omega)$ time algorithm from [26] via fast matrix multiplication, where $\omega < 2.371339$ [7]. So-called *combinatorial* algorithms, which avoid fast matrix multiplication, have achieved only sub-polynomial improvements so far despite decades-long efforts [9, 10, 15, 39, 3], leading to the conjecture that $n^{3-o(1)}$ time is unavoidable for combinatorial algorithms. A remarkable reduction by Vassilevska and Williams [38] shows that this conjecture is equivalent to the assumption that no strongly subcubic combinatorial algorithm exists for Boolean Matrix Multiplication (BMM).

▶ **Conjecture 1** (Combinatorial Triangle Detection (equivalently, the BMM Conjecture))**.** *There is no strongly subcubic combinatorial Triangle Detection (equivalently, BMM) algorithm.*

Triangle Detection is one of the most fundamental problems in algorithmic graph theory and fine-grained complexity; not only is it the smallest non-trivial case of k-Clique Detection and k-Cycle Detection, making it arguably the simplest graph problem we cannot solve in linear time, but its hardness also explains the complexity of a vast number of other problems. Indeed, many of the central conjectures in fine-grained complexity, which form the basis for dozens of conditional lower bounds, can be viewed as variants, strengthenings, or generalizations of the assumption that Triangle Detection is a hard problem. In particular, as discussed by Abboud et al. [3, Section 1.1], a refutation of the Combinatorial Triangle Detection conjecture would be a major step towards refuting central conjectures such as All-Pairs Shortest-Paths (equivalent to the weighted variant), 3SUM (related to the reporting variant), and Online Matrix Vector Multiplication (related to the online variant). This motivates the study of this particular problem for two main reasons. First, for the purpose of gaining a better understanding of what makes problems hard, e.g., informing us about the hard instances of all these problems. Second, for the purpose of attacking the conjectures of fine-grained complexity, which should be viewed as crisply articulating the edges of our algorithmic capabilities, and hence being the most pressing for algorithmic research.

An inspiring approach towards attacking Conjecture 1 is via the regularity-based techniques in the $n^3/\Omega(\log^{2.25} n)$ time algorithm of Bansal and Williams [10] and in the recent $n^3/2^{\Omega(\log^{1/7} n)}$-time algorithm of Abboud et al. [3]. Both works employ the "structure vs. randomness" methodology [35], which exploits a dichotomy between structure and pseudorandomness by decomposing the input into a structured part and a pseudorandom part. The precise definitions of structure and pseudorandomness can vary, as worst-case instances of Triangle Detection can be highly structured under one definition while being highly pseudorandom under another. For example, the neighborhood of every vertex in a triangle-free graph is an independent set (a highly structured property). However, these graphs can also be good expanders (a pseudorandom property) [4, 5]. The currently known structural properties of worst-case instances (e.g., large independent sets) are not known to be algorithmically useful for recognizing triangle-free graphs efficiently. A challenge towards algorithmic progress is in finding the right notion of pseudorandomness for this context.

In this work, we aim to improve the understanding of the structure of worst-case Triangle Detection instances. A better understanding of this question is crucial for such a fundamental problem and may direct us toward resolving Conjecture 1. We initiate the methodological study of combinatorial Triangle Detection algorithms for structured inputs. In the language of algorithmic graph theory, we consider the following question.

Which families of graphs admit a strongly subcubic combinatorial Triangle Detection algorithm?

For graph families in which every graph is sparse, namely, has $m = O(n^{2-\varepsilon})$ edges for some $\varepsilon > 0$, the answer is trivial; a straightforward $O(mn)$ algorithm is already strongly subcubic. Such graph families include bounded-arboricity graphs, minor-free graphs, bounded treewidth graphs, and more. In this paper, we focus on the family of H-free graphs for a fixed pattern H: graphs that do not contain the pattern H as a subgraph. This family of graphs is an excellent candidate for studying the properties of structured inputs. For example, H-free graphs can be dense, which implies that the following question cannot be answered by only exploiting sparsity.

▶ **Open Question 1.** *Which patterns H admit a strongly subcubic combinatorial Triangle Detection algorithm for H-free graphs?*

H-free graphs are also central objects in extremal graph theory, and the number of copies of a pattern H is often used as a measure of pseudorandomness. In particular, the Chung-Graham-Wilson theorem [18] shows that a graph in which the count of every pattern H is as expected in a random graph with the same density (up to an additive error) satisfies many equivalent notions of pseudorandomness, including strong spectral expansion. It is therefore natural to ask whether a graph containing few copies of H has enough structure to be exploited for recognizing triangle-free graphs efficiently. Throughout the paper, we use k to denote the number of vertices in H.

▶ **Open Question 2.** *For which patterns H is there a combinatorial Triangle Detection algorithm such that if a graph G contains at most $n^{k-\varepsilon}$ copies of H for some $\varepsilon > 0$, the algorithm runs in at most $n^{3-\delta}$ time for some $\delta := \delta(\varepsilon) > 0$?*

In this paper, we answer both of these questions for large and complex families of patterns. We begin with the first question about H-free graphs, which are the main focus of the paper.

1.1 Our results for Triangle Detection in H-free graphs

On the lower bound side, we identify two classes of patterns for which, under Conjecture 1, there is no strongly subcubic Triangle Detection algorithm for H-free graphs. Formally, we prove the following theorem using standard self-reductions in this branch of research.

▶ **Theorem 2** (Lower bounds classes). *If H is not 3-colorable, or if it contains more than one triangle, then combinatorial Triangle Detection algorithms for H-free graphs require $n^{3-o(1)}$ time under Conjecture 1.*

On the upper bound side, we employ three different techniques to solve the problem for different classes of patterns, putting us close to proving a *dichotomy theorem*: a complete classification of patterns into those for which Triangle Detection in H-free graphs is strongly subcubic, and those for which a conditional lower bound of $n^{3-o(1)}$ exists.

Before we present the results, let us build some intuition. Our starting point is bipartite patterns. Suppose H is a bipartite graph with parts of sizes $s \leq t$. By the Kovári-Sós-Turán theorem [33] (abbreviated as KST), an H-free graph G contains at most $O(n^{2-1/s})$ edges, so the problem can be solved trivially in $O(n^{3-1/s})$ time.

For non-bipartite graphs, the smallest case is $H = K_3$, which is trivial: If G is triangle-free, an algorithm can answer NO in constant time. Another simple case is when H is a triangle with an attached edge. In an H-free graph, any vertex of a triangle cannot have

a neighbor outside that triangle. This implies that the vertices of a triangle in G have degree 2. Therefore, one can find a triangle in $O(m+n)$ time by processing only the vertices of degree 2. The smallest non-trivial case is when H is a cycle of length 5, denoted by C_5. Our starting point is a combinatorial and strongly subcubic Triangle Detection algorithm for C_5-free graphs.

▶ **Proposition 3.** *There is a combinatorial Triangle Detection algorithm for C_5-free graphs, running in time $\widetilde{O}(n^{7/3})$ and succeeding with probability $1 - O(1/n)$.*

In later sections, we will see that this algorithm is not optimal; however, it serves as an important warm-up because it utilizes many of the core ideas in our most general results. Specifically, the approach is to try to embed the forbidden pattern randomly in the graph G. The randomness ensures that a triangle that is incident to a high-degree vertex is contained in the area surrounding the embedded pattern. Eventually, we gain a speed-up by restricting the search to an area that, on the one hand, contains a triangle edge. On the other hand, every non-triangle edge in it completes the embedding of the forbidden pattern. Essentially, the problem is reduced to deciding if there is an edge in a subgraph of G.

By generalizing this embedding approach, we provide an algorithm that applies to more patterns. It turns out that there is a large and rich family of patterns that can be embedded in worst-case Triangle Detection graphs, resulting in a polynomial speed-up. These *embeddable* patterns are characterized by admitting a special type of 3-coloring, defined as follows.

▶ **Definition 4** (Nice colorings). *Let $C : V(H) \mapsto \{\texttt{RED}, \texttt{BLUE}, \texttt{GREEN}\}$ be a proper 3-coloring of H. We say that C is* nice *if, by repeatedly deleting vertices with monochromatic neighborhoods until no such vertices remain, the leftover graph is either an empty graph or a triangle. We may also say that H is nicely colored by C.*

For patterns that admit a nice coloring, we provide the following result.

▶ **Theorem 5.** *For every pattern H that admits a nice coloring, there is a combinatorial Triangle Detection algorithm for H-free graphs, running in time:*

$$\begin{cases} \widetilde{O}(n^{3-\frac{1}{2^{k-3}}}) & \text{if } H \text{ is triangle-free,} \\ \widetilde{O}(n^{3-\frac{1}{2^{k-4}}}) & \text{if } H \text{ contains a triangle.} \end{cases}$$

The algorithm succeeds with probability $1 - O(1/n)$.

An immediate question that arises is about the generality of this result, i.e., which patterns admit a nice coloring. Let us begin by showing that it captures the most basic cases already discussed. First, observe that all bipartite graphs admit a nice coloring, because every neighborhood in a proper 2-coloring must be monochromatic. A triangle clearly admits a nice coloring. Next, observe that every cycle of odd length greater than 3 admits a nice coloring, e.g., by coloring it greedily with two colors until the process is stuck at the last vertex, which gets a third color. The reason this coloring is nice is that the second vertex to receive a color will have a monochromatic neighborhood. After removing it, the remaining pattern is a path, and its endpoints must also have a monochromatic neighborhood. As an immediate consequence, we also get that all patterns which are C_k-colorable[1] for some $k > 3$ admit a nice coloring, as the greedy coloring can also nicely color the color classes in

[1] A graph G is H-colorable if there is a homomorphism from G to H. Equivalently, it means that G is a subgraph of a blowup of H, i.e., the graph obtained by replacing every edge of H with a biclique.

a C_k-colored pattern. However, the result is even more general than C_k-colored patterns: For example, the smallest triangle-free patterns that are not C_5-colorable have 8 vertices [23, Theorem 1.1, Figure 3]. The graph in Figure 1 is such a pattern, yet it also admits a nice coloring that we illustrate in the figure.

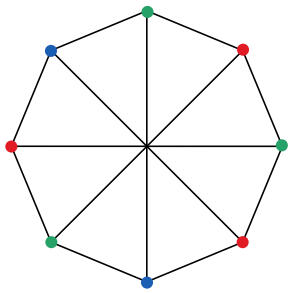

Figure 1 An 8-vertex triangle-free graph that is not C_5-colorable, but admits a nice coloring (illustrated).

Perhaps every pattern H which does not fall in the lower bound classes of Theorem 2 admits a nice coloring? From a local view, it seems plausible: If H is 3-colorable and contains at most one triangle, then the neighborhoods of non-triangle vertices are independent sets. Thus, these neighborhoods can be properly colored with only one color. A similar notion of colorings, called adynamic colorings, was studied by Šurimová et al. [34]. This paper refutes that idea by giving an example of a triangle-free, 3-colorable graph H, such that every proper 3-coloring of H does not admit a vertex with a monochromatic neighborhood. Specifically, H consists of two copies of a 6-cycle such that each vertex in the first copy is connected to two vertices in the second copy: to its copy and to the antipodal vertex of its copy. See Figure 2.

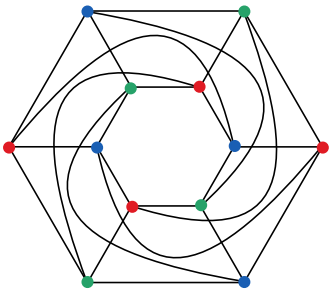

Figure 2 The graph from [34, Figure 6], together with a proper 3-coloring (illustrated). This graph does not admit a nice coloring, and in every proper 3-coloring of it, the neighborhood of every vertex is two-colored.

A curious observation that we make is that cycles of length divisible by 3 are inherent in such examples. In particular, if a 3-colorable graph is free from such cycles, then not only does it admit a nice coloring, but *every* proper 3-coloring is nice.

▶ **Proposition 6.** *Let H be a 3-colorable, $3k$-cycle free graph for every $k \in \mathbb{N}$. Then in every proper 3-coloring of H there is a vertex with a monochromatic neighborhood.*

Understanding these examples is crucial to our goal of proving a dichotomy theorem. It could be that the missing component for such a theorem is a lower bound for patterns that do not admit a nice coloring. It turns out that the story does not end here, as there are patterns H which do not admit a nice coloring, yet H-free graphs admit a strongly subcubic combinatorial Triangle Detection algorithm.

Our next family of patterns for which we show a strongly subcubic algorithm is a generalization to patterns that are essentially characterized by a single triangle being the only obstacle to a nice coloring.

▶ **Theorem 7.** *Suppose that H satisfies the following property. There is a 3-coloring of H, such that after removing s vertices with monochromatic neighborhood until no such vertices remain, the leftover pattern consists of a triangle xyz and a subpattern H', with $xy \in E(H')$ and $z \notin V(H')$, and H' is nicely colored. Then there is a combinatorial Triangle Detection algorithm for H-free graphs that runs in time $\widetilde{O}\left(n^{3-\frac{1}{2^{k+s-1}}}\right)$. The algorithm succeeds with probability $1 - O(1/n)$.*

Together, our results provide a complete picture for small patterns.

▶ **Proposition 8.** *Every pattern H with at most 8 vertices falls into one of the following four categories:*

- *Trivially Solvable: The graph is a subpattern of a triangle, and Triangle Detection can be solved in constant time.*
- *Conditional Lower Bound: The pattern is not 3-colorable or contains more than one triangle, implying no strongly subcubic combinatorial algorithm exists under Conjecture 1.*
- *Nice Coloring: The pattern admits a nice coloring, and a strongly subcubic combinatorial algorithm is provided by Theorem 5.*
- *Attached Triangle: The pattern falls under the classification in Theorem 7, and a strongly subcubic combinatorial algorithm is provided by the theorem.*

We also suggest a minimal example of a pattern H that does not fall into any of the four categories. The pattern is constructed similarly to the one in Figure 2, but we replace one of the 6-cycles with a triangle. See Figure 3.

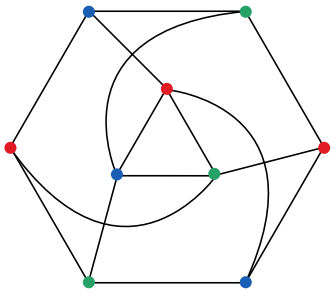

Figure 3 A minimal example of a pattern which does not fall into any of the categories of Proposition 8. Namely, the pattern is 3-colorable (illustrated) and contains exactly one triangle. In every proper 3-coloring of it, there are no vertices with a monochromatic neighborhood. Moreover, every vertex of the triangle has degree greater than 2 and therefore it does not fall into the classification of Theorem 7.

1.2 H-sensitive algorithms

Another set of results we provide addresses Open Question 2 about H-sensitive algorithms. These are algorithms that run faster when the number of copies of a pattern H is polynomially smaller than the maximum possible, which can be as high as $\Omega(n^k)$. H-sensitive algorithms are desirable because their complexity improves upon the worst-case complexity of the problem (i.e., they are never slower). In particular, an H-sensitive algorithm demonstrates

that the hardest instances of Triangle Detection contain a maximum number of copies of H, aligning with the idea that pseudorandom graphs are the most challenging. We provide a generalization of the algorithm from Theorem 5 to an H-sensitive algorithm for every H that admits a nice coloring.

▶ **Theorem 9.** *For every pattern H that admits a nice coloring, there is a combinatorial algorithm that, given a graph G and a value t which upper bounds the number of copies of H in G, decides whether G contains a triangle. The running time is given by:*

$$\begin{cases} \widetilde{O}\left(n^{3-\frac{1}{2^k-1}} + n^{3-\frac{k}{2^k-1}} \cdot t^{\frac{1}{2^k-1}}\right) & \text{if } H \text{ is triangle-free,} \\ \widetilde{O}\left(n^{3-\frac{1}{2^k-4}} + n^{3-\frac{k-3}{2^{k-3}-1}} \cdot t^{\frac{1}{2^{k-3}-1}}\right) & \text{if } H \text{ contains a triangle.} \end{cases}$$

The algorithm succeeds with probability $1 - O(1/n)$.

Note that the distinction between triangle-free patterns and those that contain a triangle is necessary: By Theorem 2, there exist worst-case Triangle Detection instances with only one triangle, and consequently, only $O(n^{k-3})$ copies of H for every pattern H that contains a triangle, as opposed to potentially $\Omega(n^k)$ copies in the triangle-free case. Therefore, an H-sensitive algorithm for a pattern H that contains a triangle should run in cubic time when $t = \Omega(n^{k-3})$, whereas if H is triangle-free, then the running time for the same value of t should be strongly subcubic.

We also remark that the algorithm requires an upper bound t on the number of copies of H as input. This assumption slightly weakens the result, but it can be removed easily in the case where H is triangle-free by computing an (not necessarily tight) upper bound on the number of copies of H during a preprocessing step, which still yields a strongly subcubic algorithm if G has polynomially fewer than n^k copies of H.[2] We note that if H contains a triangle, then it is necessary for an H-sensitive algorithm to receive t as an input, under Conjecture 1: If there was an H-sensitive Triangle Detection algorithm that is oblivious to the upper bound t, then it would need to run in subcubic time on every triangle-free graph G (as then $t = 0$). Using such an algorithm, we can distinguish triangle-free graphs from graphs that contain a triangle by running the supposed algorithm and answering YES if its running time exceeds the stated bound when $t = 0$.

Additionally, we provide a more efficient algorithm for the special case of $H = C_5$.

▶ **Theorem 10.** *There is a combinatorial Triangle Detection algorithm that runs in $\widetilde{O}(n^2 + n^{4/3}t^{1/3})$ time, where t is the number of 5-cycles in the input graph. The algorithm succeeds with probability at least $1 - O(1/n)$.*

Note that, unlike the previous algorithm, this one is oblivious to t.

1.3 A special focus on C_{2k+1}-free graphs

Cycles (using C_k to denote the cycle of length k) are one of the most natural classes of graphs, so a natural question is about the running time of Triangle Detection in C_k-free graphs, for each k. For the case of even cycles, C_{2k} is bipartite, so the KST theorem can be employed to get an $O(n^{3-1/k})$ time algorithm. In fact, it is possible to do much better than that, because

[2] For example, consider a triangle-free pattern H. Suppose that we take $\Theta(n^2 \log n)$ samples of k-tuples of vertices, and for each tuple we check if it induces a copy of H. If the first copy of H was found after $\Omega(n^\alpha \log n)$ samples, for any $0 < \alpha \leq 2$, we conclude that $t \leq n^{k-\alpha}$ with high probability. If no copy appears in $\Theta(n^2 \log n)$ samples, we conclude that $t \leq n^{k-2}$.

not only is the graph sparse, it follows from the Bondy-Simonovits theorem that a C_{2k}-free graph is d-degenerate for $d = O(n^{1/k})$ [13]. A graph is d-degenerate if every induced subgraph contains a vertex of degree less than d. The classic Chiba-Nishizeki algorithm solves Triangle Detection in d-degenerate graphs in $O(md) = O(n^{1+2/k})$ time [17].[3] For odd cycles, the problem is more complex, as illustrated by complete bipartite graphs, which are C_{2k+1}-free but also very dense. We overcome this challenge by essentially restricting the search to graphs that have the form of a dense bipartite graph with a bounded degeneracy graph added to one of its parts. This requires a generalization of the Chiba-Nishizeki algorithm to graphs that are only partially degenerate. In addition, we employ a novel idea, an algorithmic adaptation of a supersaturation result for even cycles by Jiang and Yepremyan [27]. Together, we manage to prove the following theorem.

▶ **Theorem 11.** *For every $k \geq 2$, there is a combinatorial Triangle Detection algorithm for C_{2k+1}-free graphs that runs in $\widetilde{O}(m + n^{1+2/k})$ time.*

Note that this running time matches the bound for C_{2k}-free graphs (using Chiba-Nishizeki), up to an additive m term.

A summary of the results so far is presented in Table 1.

Table 1 Our results for Triangle Detection in H-free graphs.

Classification	Condition on H	Bound	Theorem
Lower bound under the BMM Conjecture	Not 3-colorable	$n^{3-o(1)}$	2
	Contains more than one triangle	$n^{3-o(1)}$	2
Subcubic upper bound	$H = C_{2k+1}$ $(k \geq 2)$	$\widetilde{O}(m + n^{1+2/k})$	11
	Nicely colored and triangle-free	$\widetilde{O}(n^{3-\frac{1}{2^{k-3}}})$	5
	Nicely colored and contains a triangle	$\widetilde{O}(n^{3-\frac{1}{2^{k-4}}})$	5
	"Attached Triangle" $(s = \#$ removable vertices$)$	$\widetilde{O}(n^{3-\frac{1}{2^{k+s-1}}})$	7
Subcubic upper bound for any $t < n^{k-\varepsilon}$ $(t = \#H)$	$H = C_5$	$\widetilde{O}(n^2 + n^{4/3}t^{1/3})$	10
	Nicely colored and triangle-free	$\widetilde{O}(n^{3-\frac{1}{2^{k-1}}} + n^{3-\frac{k}{2^{k-1}}} \cdot t^{\frac{1}{2^{k-1}}})$	9
	Nicely colored and contains a triangle	$\widetilde{O}(n^{3-\frac{1}{2^{k-4}}} + n^{3-\frac{k-3}{2^{k-3}-1}} \cdot t^{\frac{1}{2^{k-3}-1}})$	9

1.4 Generalized Turán problem and Triangle Listing

A related line of work has considered the purely combinatorial question of bounding the number of triangles in H-free graphs. This Turán-type question was initiated in a paper by Alon and Shikhelman [8], and since its introduction, many papers on the subject have appeared (see [22] for a recent survey). For every pattern H, let $ex(K_3, H)$ be the maximum number of triangles in an n-vertex H-free graph. An analogous result to our first lower bound class in Theorem 2 follows from [8, Proposition 2.1] and shows that $ex(K_3, H)$ is strongly subcubic if and only if H is 3-colorable. One might wonder if the proof is algorithmic, in the sense that it also provides an efficient way to find the triangles if H is 3-colorable. However, an algorithmic proof of this result cannot hold generally for all 3-colorable patterns. That is because there are patterns that are 3-colorable but do not admit a strongly subcubic combinatorial Triangle Detection algorithm under Conjecture 1. For example, if H is the

[3] The original paper uses arboricity rather than degeneracy, but the two parameters are known to differ by at most a factor of two.

diamond graph (two triangles that share an edge), then clearly $ex(K_3, H) = O(n^2)$, whereas Theorem 2 shows that Triangle Detection in diamond-free graphs requires $n^{3-o(1)}$ time under Conjecture 1. In the opposite direction, the algorithmic question may contribute to the mathematical one, as a combinatorial Triangle Detection algorithm for H-free graphs offers insight into the structure of triangles in H-free graphs, which may be used to prove upper bounds on $ex(K_3, H)$.

A more relatable variant of Triangle Detection is the Triangle Listing problem, in which the goal is to produce a list of all the triangles in a graph. This problem generalizes not only Triangle Detection but also its weighted variants, such as Negative Triangle (an APSP-hard problem [38]) and 0-Triangle (a 3SUM-hard problem [37]). A primary objective in designing a listing algorithm is to ensure it is output-sensitive, meaning the running time is proportional to the number of triangles in the graph. Ideally, the running time should match the running time for detection when the output size is small and scale up to $\Theta(n^3)$ when the number of triangles reaches its maximum of $\Omega(n^3)$ (see e.g., [11]). The problem of combinatorial Triangle Listing for general graphs is straightforward, as the trivial $O(n^3)$ time algorithm for detection can also list all the triangles in the same running time and is tight under Conjecture 1. In the setting of H-free graphs, the situation is less clear. Our lower bounds for detection in Theorem 2 clearly hold for listing as well. For H that does not fall into a lower bound class, the trivial $O(n^3)$ time listing algorithm seems wasteful because the output size is at most $ex(K_3, H)$, which is strongly subcubic by [8, Proposition 2.1] (as H is 3-colorable).

A general upper bound on $ex(K_3, H)$ that applies to every 3-colorable H is given by plugging a KST-type theorem for 3-uniform hypergraphs [40, Theorem 1.5.7] into the proof of [8, Proposition 2.1], resulting in $ex(K_3, H) = O(n^{3-1/k^2})$. For the special case where $H = C_{2k+1}$, it is established that $ex(K_3, C_{2k+1}) = O(ex(K_2, C_{2k}))$ [25], where $ex(K_2, C_{2k})$ is the maximum number of edges in a C_{2k}-free graph, bounded by $O(n^{1+1/k})$ [13]. A general lower bound by Kostochka et al. [32] shows that for every 3-colorable H, $ex(K_3, H) = \Omega(n^{3-\frac{3k-9}{e(H)-3}})$, where $e(H)$ is the number of edges in H. In particular, for dense 3-colorable patterns we get $ex(K_3, H) = n^{3-O(1/k)}$. This gives a lower bound on the running time for listing, as the output size can be as large as $ex(K_3, H)$.

Our Triangle Detection algorithms for H-free graphs generalize well to the listing setting, producing a list of all triangles in the same time it takes to detect a single one. The running time is independent of the output size because, as discussed, the output size in H-free graphs is bounded by $O(n^{3-1/k^2})$ (or $O(n^{1+1/k})$ if $H = C_{2k+1}$), whereas our algorithms run slower.

▶ **Theorem 12** (Listing for nicely colored patterns). *Let H be a pattern that admits a nice 3-coloring. Then there is a combinatorial algorithm that lists all triangles in an H-free graph, in time:*

$$\begin{cases} \widetilde{O}(n^{3-\frac{1}{2^{k-3}}}) & \text{if } H \text{ is triangle-free,} \\ \widetilde{O}(n^{3-\frac{1}{2^{k-4}}}) & \text{if } H \text{ contains a triangle.} \end{cases}$$

▶ **Theorem 13** (H-sensitive listing for nicely colored patterns). *For every pattern H that admits a nice coloring, there is a combinatorial algorithm that, given a graph G and a value t which upper bounds the number of copies of H in G, lists all the triangles in the graph. The running time is given by:*

$$\begin{cases} \widetilde{O}\left(n^{3-\frac{1}{2^{k-1}}} + n^{3-\frac{k}{2^k-1}} \cdot t^{\frac{1}{2^k-1}}\right) & \text{if } H \text{ is triangle-free,} \\ \widetilde{O}\left(n^{3-\frac{1}{2^{k-4}}} + n^{3-\frac{k-3}{2^{k-3}-1}} \cdot t^{\frac{1}{2^{k-3}-1}}\right) & \text{if } H \text{ contains a triangle.} \end{cases}$$

▶ **Theorem 14** (Listing for patterns with an attached triangle). *Suppose that H satisfies the following property. There is a 3-coloring of H, such that after removing s vertices with monochromatic neighborhoods until no such vertices remain, the leftover pattern consists of a triangle xyz and a subpattern H', with $xy \in E(H')$ and $z \notin V(H')$, and H' is nicely colored. Then there is a combinatorial algorithm that lists all triangles in an H-free graph in time $\widetilde{O}\left(n^{3-\frac{1}{2^{k+s-1}}}\right)$. The algorithm succeeds with probability $1 - O(1/n)$.*

▶ **Theorem 15** (Listing for odd cycles). *For every $k \geq 2$, there is a combinatorial algorithm that lists all triangles in a C_{2k+1}-free graph in $\widetilde{O}(m + n^{1+2/k})$ time.*

In the case of nicely colored patterns, we observe a significant gap between $ex(K_3, H) = O(n^{3-1/k^2})$ and the stated running time. It is an interesting question whether the exponential dependence in our running time could be replaced by, e.g., a polynomial dependence $O(n^{3-1/k^{O(1)}})$. In the case of odd cycles, we see a smaller gap, a factor roughly $n^{1/k}$ between the combinatorial bound and the algorithmic running time. It seems likely that some algorithmic overhead is necessary for finding the triangles, even in C_{2k+1}-free graphs.

Paper organization

We provide a high-level presentation of our core ideas in Section 2, followed by a discussion of related work in Section 3. We then defer to the full version of the paper for the complete proofs. We conclude with summary and open questions in Section 4.

2 Technical Overview

In this section, we present the core ideas of the algorithms in this paper. We begin with the main result, which is the embedding algorithm.

2.1 The embedding approach

We begin with a simple $\widetilde{O}(n^{7/3})$-time algorithm for C_5-free graphs, demonstrating the techniques and proving Proposition 3.

Proof of Proposition 3. Let $1 \leq \Delta \leq n$ be a degree threshold to be set later. For every vertex with a degree smaller than Δ, we can check if it participates in a triangle in $O(\Delta^2)$ time by examining all pairs of its neighbors. The algorithm begins by removing all vertices of degree less than Δ in a total of $O(n\Delta^2)$ time, until the remaining graph has a minimum degree of at least Δ. Once the graph has a high minimum degree, a random sample of $\Theta(\frac{n}{\Delta} \log n)$ vertices will hit the neighborhood of a triangle vertex (if one exists) with high probability. We take such a sample and check whether any sampled vertex is itself a triangle vertex. This takes $O(n^2)$ per vertex for a total of $\widetilde{O}(\frac{n^3}{\Delta})$ time. If no triangle is found, we proceed by processing the radius-2 balls around each of the sampled vertices, for a total of $\widetilde{O}(\frac{n^3}{\Delta})$ time (as the number of edges in each ball could be as high as $\Theta(n^2)$). Observe that one of those balls contains the triangle with high probability. Now suppose that v is a sampled vertex whose ball contains a triangle xyz, and let $N_1(v)$ and $N_2(v)$ denote the vertices at distances 1 and 2 from v, respectively. Since v is not in a triangle (guaranteed above), there must be exactly one triangle vertex, say x, in $N_1(v)$, while the other two vertices, y and z, are in $N_2(v)$. The approach is to search for such a triangle by examining every edge in $N_2(v)$ and a vertex in $N_1(v)$. Naively, this could take $O(n^3)$ time for each ball.

To exploit the C_5-free property, observe that for any edge uw within $N_2(v)$, if the "parents" of its endpoints (i.e., neighbors of u and w in the upper layer $N_1(v)$) $p_w, p_u \in N_1(v)$ are distinct, a C_5 is formed with v. Since the graph is C_5-free, any two parents of u and w must be identical. This implies that u and w share a common parent, say $p \in N_1(v)$, which forms a triangle puw. Therefore, a triangle exists if and only if there is an edge within $N_2(v)$. This can be verified in $O(n^2)$ time per ball, for a total of $\widetilde{O}(n^3/\Delta)$ time. The overall running time is $\widetilde{O}(n\Delta^2 + n^3/\Delta)$, which is optimized to $\widetilde{O}(n^{7/3})$ by setting $\Delta = n^{2/3}$. ◀

This algorithm for C_5-free graphs works by attempting to embed a C_5 in a C_5-free graph G. It gains a speed-up by restricting the search to a region of the graph where: (1) a triangle edge is highly likely to exist, and (2) any edge not part of a triangle would necessarily form a C_5. The goal is to generalize this approach to work for general patterns, but it is unclear what other patterns can be embedded. Hence, we will provide a general framework for this embedding approach, which will guide us in characterizing embeddable patterns. As mentioned earlier, we identify a broad class of patterns amenable to this embedding approach: those that admit a nice coloring, whose definition is restated below.

▶ **Definition 4** (Nice colorings). *Let $C : V(H) \mapsto \{\texttt{RED}, \texttt{BLUE}, \texttt{GREEN}\}$ be a proper 3-coloring of H. We say that C is* nice *if, by repeatedly deleting vertices with monochromatic neighborhoods until no such vertices remain, the leftover graph is either an empty graph or a triangle. We may also say that H is nicely colored by C.*

Suppose that a graph G contains a triangle xyz, and suppose for simplicity that it is the only triangle in the graph (in Section 5 of the full version we show that we can assume so without loss of generality). An embedding algorithm takes a pattern H and embeds its vertices one at a time in G. At stage i, for $i = 1, 2, \ldots, k$, a vertex $h_i \in V(H)$ is mapped to a vertex $v_i \in V(G)$. The basic requirements for an embedding algorithm are as follows:

- For each $i \in [k]$, there exists some $t_i \in \{x, y, z\}$ such that $v_i \in N(t_i)$. This ensures that the area surrounding the embedded pattern contains the triangle.
- For each $i \in [k]$, v_i lies in the common neighborhood of all the previously embedded neighbors of h_i. Formally,

$$\forall i \in [k]: \quad v_i \in \bigcap_{\substack{j<i \text{ such that} \\ h_i h_j \in E(H)}} N(v_j).$$

This ensures that the embedding respects the edges of H.

An important observation is that for every $i \in [k]$, t_i must be unique, i.e., v_i can only be adjacent to a single vertex in $\{x, y, z\}$. That is because otherwise, there would be another triangle in G containing v_i and two vertices in $\{x, y, z\}$. Moreover, the following requirement from the algorithm follows: For every neighbor h_j of h_i, we must have $t_i \neq t_j$. That is because v_j and v_i are adjacent (by the second requirement), so if $t_i = t_j$, there would be another triangle $v_i v_j t_i$ in G. This implies that the patterns that the algorithm can embed must be 3-colorable: let the color of h_i be t_i for every $i \in [k]$.

Note that there is no guarantee that the intersection set from the second bullet above is non-empty. If it becomes empty at some stage, then the embedding algorithm gets stuck. To have this guarantee, we add another requirement from the embedding algorithm. The requirement is that for each $i \in [k]$ and every two neighbors of h_i, h_j and h_ℓ, where $j, \ell < i$, the triangle neighbors of v_j and v_ℓ will be the same. Formally:

$$\forall i \in [k]: \quad \forall^{j,\ell<i}_{h_j, h_\ell \in N_H(h_i)}: \quad t_j = t_\ell.$$

This ensures that the set $N(v_j) \cap N(v_\ell)$ is non-empty, as it contains a triangle vertex $t_j = t_\ell$. In particular, the intersection set from the second bullet is non-empty because it contains a unique triangle vertex. Note that the 3-coloring of H we now get is a nice coloring: Starting from the last vertex in the embedding, every vertex has a monochromatic neighborhood in H.

An algorithm for nicely colored patterns

We are now ready to present the embedding algorithm from Theorem 5 for patterns that are nicely colored. Suppose that H is nicely colored and that G is 3-colored (in Section 5 of the full version we show that this assumption is without loss of generality), so in particular, G is free from nicely colored copies of H. The algorithm maintains the invariant of a high min-degree to every color class. Namely, suppose that some vertex (e.g., a red vertex) contains fewer than Δ neighbors in some other color class (e.g., the blue vertices). In that case, it can be verified in $O(n\Delta)$ time that the vertex does not participate in a triangle, resulting in a total of $O(n^2\Delta)$ time for a low-degree cleanup. In each stage, the vertex of H whose neighborhood is monochromatic is embedded by taking a random sample of $\widetilde{O}(n/\Delta)$ vertices from its color class in G. Then, the color class of G corresponding to its monochromatic neighborhood is restricted to the neighborhoods of the sampled vertices (creating $O(n/\Delta)$ different subgraphs of G). Because of high minimum degrees, at least one sampled vertex is guaranteed to hit the neighborhood of a triangle vertex, so the resulting subgraph still contains the triangle. As the process continues, the color classes of G are restricted to neighborhoods of embedded vertices. The triangle is found when it becomes incident to a low-degree vertex, which will surely happen once a base case is reached. We remark that after each stage, a new low-degree cleanup needs to be applied to every subgraph. Applying this cleanup naively to each of the $\widetilde{O}(n/\Delta)$ new subgraphs would take cubic $\widetilde{O}(\frac{n}{\Delta} \cdot n^2\Delta)$ time. Instead, a decreasing sequence of thresholds $\Delta_k > \Delta_{k-1} > \cdots > \Delta_1$ is introduced, and a different threshold is used in each recursive level. The final running time is obtained by optimizing them.

The case of an attached triangle

Let us now explain the algorithm from Theorem 7. The idea is to first embed the vertices with monochromatic neighborhoods, as in the case of nicely colored patterns. Then, we reach the hard-core of the pattern: a nicely colored subpattern with an attached triangle on one of its edges. For this pattern, the key idea is to sparsify the graph by reducing the number of copies of the subpattern H'. Namely, every edge that participates in many colored copies of H' is removed. Once the number of copies of H' becomes polynomially smaller than $n^{|V(H')|}$, we apply an H'-sensitive algorithm as a subroutine to the remaining graphs.

2.2 The approach for H-sensitive algorithms

It is possible to generalize the embedding algorithm to be H-sensitive by employing the following idea: In a sample of $\widetilde{O}(n/\Delta)$ vertices in a graph with high-minimum degree and at most t copies of H, one of the sampled vertices is an *H-light* neighbor of the triangle with high probability. That is, a neighbor of the triangle that participates in at most $O(t/\Delta)$ copies of H. This ensures that the number of copies of the forbidden pattern reduces by a factor of Δ at each step. When a base case is reached, the number of copies of a very basic pattern (e.g., a single vertex) is small enough for the base case to be solved efficiently. For instance, if the number of vertices of a certain color is small, triangles incident to them can be found quickly.

A specialized C_5-sensitive algorithm

We now explain how to improve the running time of the previously discussed $\widetilde{O}(n^{7/3})$-time algorithm for C_5-free graphs to $\widetilde{O}(n^2)$ by changing a single component: Instead of cleaning up low-degree vertices and then focusing on the remaining graph that has high min-degree, we guess the maximum degree of a vertex in a triangle, and then ignore vertices with higher degrees in G. The guess can be implemented by bucketing, i.e., trying a sequence of exponentially increasing guesses. We then follow the same steps as in the $O(n^{7/3})$ time algorithm: Sample $\widetilde{O}(n/\Delta)$ vertices and search for triangles in the 2-hop ball around each sampled vertex. However, instead of constructing a 2-hop ball around each vertex, we only process vertices of degree at most Δ. The time to process each ball becomes $O(n\Delta)$, for a total of $\widetilde{O}(\frac{n}{\Delta} \cdot n\Delta) = \widetilde{O}(n^2)$ time. To obtain a C_5-sensitive algorithm, we relate the search time for a triangle inside a random 2-hop ball to the number of 5-cycles in that ball. Using the fact that the expected number of 5-cycles incident to a random vertex is $O(t/n)$, we get the bound of $\widetilde{O}(n^2 + n^{4/3}t^{1/3})$.

2.3 The approach for C_{2k+1}-free graphs

Let us now explain the high-level overview of the algorithm for C_{2k+1}-free graphs. Unlike the previous algorithms, this one is deterministic and employs a significantly different approach. The idea is to process the layers of a k-hop ball around a vertex as long as the layers are expanding. The algorithm searches for triangles in the first $k-1$ layers using a generalized Chiba-Nishizeki algorithm. If a triangle is not found, then most of the ball is deleted (except for the last two layers, which may still contain a triangle), ensuring a bounded number of phases before we run out of vertices. This approach exploits the structure of C_{2k+1}-free graphs using three main components:

- A lemma stating that the subgraph induced by any single layer in the ball is $O(k)$-degenerate.
- A generalization of the Chiba-Nishizeki algorithm that can find triangles with at least one edge in a specified $O(k)$-degenerate subgraph. This algorithm is useful for finding a triangle with an edge that lies within a layer.
- An efficient algorithm that identifies a large fraction of edges between the last two layers of the ball that participate in a $2k$-cycle, and for each such edge, it finds a $2k$-cycle containing it.

The last component is crucial: An edge that participates in a $2k$-cycle cannot form a triangle with a vertex outside that cycle without creating a $(2k+1)$-cycle. Thus, to check such an edge for triangles, one only needs to inspect the adjacencies to the other $2k-2$ vertices of a known $2k$-cycle containing it, which takes $O(k) = O(1)$ time.

3 Related Works

3.1 Complexity dichotomies in H-free graphs

A large body of research is devoted to proving *dichotomy theorems* for H-free graphs: showing a separation between patterns for which a problem is "hard" in H-free graphs and patterns for which the problem becomes "easy". The precise definitions of hard and easy may vary, but most of this research uses coarse notions of hardness, such as NP-completeness versus polynomial-time. For example, such theorems have been proven for Max-Cut, Maximum Independent Set, and List-Coloring [30, 6, 24]. An almost complete dichotomy also exists for Subgraph Isomorphism [12].

Interestingly, a comprehensive work by Johnson et al. [29] provided a meta-theorem that captures many problems for which such a dichotomy exists, using a framework applicable in different complexity regimes. Roughly speaking, the meta-theorem states that if a problem (e.g., Maximum Independent Set) is easy on bounded-treewidth graphs, hard on subcubic graphs (graphs of degree at most 3), and hard on edge subdivisions of subcubic graphs, then a complete dichotomy exists between patterns for which the problem is easy and patterns for which it is hard in H-free graphs. Their framework is applied to various problems, primarily in the coarse complexity regimes. However, the paper also demonstrates its generality by applying it to problems in P. Namely, they consider the Diameter and Radius problems, showing a separation between patterns for which the problem is solvable in near-linear time and patterns for which the problem does not admit a strongly subquadratic algorithm.

Our work uses the hard and easy notions of cubic and strongly subcubic time, respectively. Under these notions, Triangle Detection does not fall into their framework because the problem is solvable in linear time on subcubic graphs, which makes it easy. One may wonder about the status of the NP-complete and parameterized generalizations of Triangle Detection, i.e., Maximum Clique and k-Clique. For Maximum Clique, the problem in H-free graphs is trivially solvable in polynomial time for any fixed H, as the largest clique in an H-free graph has size less than $|H|$. For k-Clique, one could seek an FPT/W[1] dichotomy between patterns. This problem is interesting only when $|H| > k$, but it appears not to have been considered in the literature yet.

The induced case

A similar and equally important question, which we do not address in this work, concerns the complexity of Triangle Detection in *induced* H-free graphs. Every H-free graph is also an induced H-free graph, but the opposite does not necessarily hold. This implies that dichotomies in the two settings differ; for example, a pattern H could exist for which the problem is hard on induced H-free graphs but easy in H-free graphs. The induced setting also behaves more nicely with respect to graph inversion because a graph G is induced H-free if and only if its complement $\bar{G}$ is $\bar{H}$-free (a similar behavior does not hold in the H-free setting). Inverting the graph is useful when considering the generalizations of Triangle Detection, like k-Clique and Maximum Clique, because of the duality between these problems and their Independent Set counterparts. The Maximum Independent Set problem on induced H-free graphs is very well-studied, and a recent work established a dichotomy for Maximum Independent Set between patterns for which the problem is NP-complete and those for which it is in quasipolynomial time [21]. Whether this extends to a P/NP dichotomy remains a major open question. In the parameterized setting, one paper has made progress toward an FPT/W[1] dichotomy for the k-Independent-Set problem [14].

3.2 Distance oracles and short cycle removal

The approximate distance oracle approach

An alternative approach for graphs that are $C_{2\ell+1}$-free for every $\ell \in [2, k]$ is to use approximate distance oracles: a data structure that upon a query (u, v) returns an approximate distance $\text{dist}(u, v) \leq \widetilde{d} \leq t \cdot \text{dist}(u, v)$ for some *stretch* $t \geq 1$. Suppose that G is a tripartite graph with parts A, B, C, and we delete all edges in $E(B, C)$. A distance query for a deleted edge $uv \in E(B, C)$ can determine whether uv participates in a triangle in G: $\text{dist}(u, v) = 2$ if and only if uv is part of a triangle. If G is $C_{2\ell+1}$-free for all $\ell \in [2, k]$, then for non-triangles,

$\text{dist}(u, v)$ is an odd number strictly greater than $2k + 1$. Hence, an approximate distance oracle with an appropriate stretch can distinguish between the YES and NO cases. For example, the Thorup-Zwick oracle [36] with a stretch t given by:

$$t = \begin{cases} k/2 & k \text{ even} \\ (k+1)/2 & k \text{ odd} \end{cases}$$

distinguishes between these cases and runs in time $O(mn^{1/t})$. This improves upon our algorithm from Theorem 11 only for odd k and for graphs with $m = O(n^{1+\frac{2}{k(k+1)}})$ edges. Moreover, this technique does not address graphs that are only C_{2k+1}-free, whereas our approach in Theorem 11 does.

Short cycle removal

A paper by Abboud et al. [2] shows that for $\Theta(\sqrt{n})$-regular graphs, results opposite to ours hold. Roughly speaking, they show a self-reduction that reduces the number of k-cycles from a maximum of $n^{k/2+1/2}$ to $O(n^{k/2+\gamma})$ for some $\gamma < 1/2$, without removing triangles. Furthermore, they show a self-reduction that removes all copies of C_4 from a graph with maximum degree $O(\sqrt{n})$ without removing any of its triangles. This powerful technique provides lower bounds for approximate distance oracles, 4-Cycle Detection, and Triangle Detection in C_4-free graphs. Subsequent works [1, 28] improved these results via 3-SUM reductions that output worst-case graphs with fewer than the maximum number of k-cycles. This reveals a surprising dichotomy: While our algorithm from Theorem 9 gains a significant speed-up on dense graphs with fewer than the maximum number of copies of a pattern H, in the sparse regime, worst-case instances do not necessarily have the maximum number of copies of the pattern. In fact, the result for 4-cycles shows that worst-case instances of Triangle Detection can even have fewer than the *average* number of C_4's in a random $\sqrt{n}$-regular graph.

The current best lower bound for the running time of approximate distance oracles is from [1], but the stretch they obtain is still a factor of 2 shy of the stretch from the Thorup-Zwick oracle for the same running time. One might try to improve upon the self-reduction technique of [2], for example, by picking different density regimes or by attempting to remove only short odd cycles instead of all short cycles. The algorithm from Theorem 11 places a limit on what can be proven with such techniques, as it runs faster than the Thorup-Zwick oracle for almost every m and k.

3.3 Additional works

The paired pattern problem

Dalirrooyfard et al. [20] considered a paired pattern problem that asks to detect whether a graph contains at least one of two patterns. For example, the $\{K_3, H\}$-Detection problem asks to detect whether a graph contains a triangle or a copy of H. An algorithm for $\{K_3, H\}$-Detection clearly solves the Triangle Detection problem in H-free graphs. However, the opposite does not necessarily hold. Still, one may ask whether our embedding approach solves the paired pattern problem as well. The answer is negative, because our algorithm relies on deleting low-degree vertices after it verifies that they do not participate in a triangle. However, those vertices may participate in copies of H, and the algorithm will fail to detect them.

Listing large cliques from a few small cliques

Another result by Dalirrooyfard et al. [19] for k-Clique Listing is an algorithm whose running time is sensitive to the number of ℓ-cliques for $\ell < k$. This result resembles our H-sensitive algorithms but in the opposite direction: the pattern with few copies is a subpattern of the target pattern. In contrast, our work considers patterns that are larger than a triangle and do not necessarily contain one.

Triangle Detection in dense graph families

Several papers have considered Triangle Detection in various graph families, resulting in very efficient combinatorial algorithms. The only such families that are also dense that we are aware of are geometric intersection graphs. A work by Kaplan et al. [31] shows a near-linear time algorithm for disk graphs, and a recent work by Chan [16] shows strongly subcubic algorithms for intersection graphs of more objects: boxes, line segments, fat objects, and translates. While these algorithms use fast matrix multiplication, they remain subcubic even when using naive matrix multiplication.

4 Conclusions

Our work is a first step in the systematic study of structured instances of combinatorial Triangle Detection. It makes progress towards a dichotomy theorem for H-free graphs, with all patterns of size at most 8 classified. Several open questions arise. The most pressing one is to complete the dichotomy for all patterns. We put forth the following hypothesis:

▶ **Hypothesis** (Dichotomy for Triangle Detection in H-Free Graphs). *The Triangle Detection problem in H-free graphs is:*

- *in $O(n^{3-\varepsilon})$ time via a combinatorial algorithm for some $\varepsilon > 0$, if H is 3-colorable and contains at most one triangle; and*
- *as hard as Triangle Detection in general graphs, otherwise.*

Note that this hypothesis is making two non-obvious claims: First, that there is a dichotomy by which each pattern H can be classified as either "easy" or *triangle-hard*; i.e., that there is no H for which Triangle Detection in H-free graphs requires cubic time but there is no reduction from Triangle Detection in general graphs to Triangle Detection in H-free graphs. Second, it asserts that the only two conditions that can make a pattern triangle-hard are having a chromatic number greater than 3 or having more than one triangle.

To prove the hypothesis, one has to extend our algorithms to more patterns, and to disprove it, one has to extend our lower bounds. We believe that the former is more likely. In fact, our current algorithmic techniques of "embedding" subpatterns by sampling vertices may be sufficient if new, more intricate analysis tools are developed.

Other open questions

From the lower bound perspective, our current understanding is that some patterns admit an $n^{3-o(1)}$ lower bound, while any other pattern only admits the trivial $\Omega(n^2)$ lower bound. It would be interesting to prove a super-quadratic lower bound (i.e., $\Omega(n^{2+\varepsilon})$ for some $\varepsilon > 0$) for Triangle Detection in H-free graphs for some pattern H that admits a subcubic algorithm. This also addresses the goal of optimizing the running times, which was not our main focus in the paper.

Another open question is about the necessity of randomness in our algorithms. In the H-free setting, it seems possible to avoid randomness by using deletions, as is done in our specialized algorithm for C_{2k+1}-free graphs. However, in the H-sensitive setting, our algorithm crucially depends on randomness as it needs to quickly find a neighbor of the triangle that participates in a small number of copies of H. Derandomizing this component seems more difficult.

Finally, this work may be further developed, for example, by considering k-Clique Detection in H-free graphs or directed versions of the problem. Another equally important question is about Triangle Detection in induced H-free graphs, which we did not address in this work.

References

1 Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-sum lower bounds for approximate distance oracles via additive combinatorics. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 391–404, 2023. doi:10.1145/3564246.3585240.

2 Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in P via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1487–1500. ACM, 2022. doi:10.1145/3519935.3520066.

3 Amir Abboud, Nick Fischer, Zander Kelley, Shachar Lovett, and Raghu Meka. New graph decompositions and combinatorial boolean matrix multiplication algorithms. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 935–943, 2024. doi:10.1145/3618260.3649696.

4 Amir Abboud and Nathan Wallheimer. Worst-case to expander-case reductions. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, pages 1:1–1:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.1.

5 Amir Abboud and Nathan Wallheimer. Worst-case to expander-case reductions: Derandomized and generalized. In *32nd Annual European Symposium on Algorithms (ESA 2024)*, pages 4:1–4:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.4.

6 Vladimir Evgen'evich Alekseev and DV Korobitsyn. Complexity of some problems on hereditary classes of graphs. *Diskretnaya Matematika*, 4(4):34–40, 1992.

7 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.

8 Noga Alon and Clara Shikhelman. Many t copies in h-free graphs. *Journal of Combinatorial Theory, Series B*, 121:146–172, 2016. doi:10.1016/J.JCTB.2016.03.004.

9 Vladimir L Arlazarov, EA Dinic, MA Kronrod, and IA Faradzev. On economical construction of the transitive closure of a directed graph. In *Dokl. Akad. Nauk SSSR*, volume 194, pages 1209–1210, 1970.

10 Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. *Theory Comput.*, 8(1):69–94, 2012. doi:10.4086/toc.2012.v008a004.

11 Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *International Colloquium on Automata, Languages, and Programming*, pages 223–234. Springer, 2014. doi:10.1007/978-3-662-43948-7_19.

12 Hans L Bodlaender, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Yoshio Okamoto, Yota Otachi, and Tom C van der Zanden. Subgraph isomorphism on graph classes that exclude a substructure. *Algorithmica*, 82(12):3566–3587, 2020. doi:10.1007/S00453-020-00737-Z.

13 John A Bondy and Miklós Simonovits. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B*, 16(2):97–105, 1974.

14 Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Stéphan Thomassé, and Rémi Watrigant. Parameterized complexity of independent set in h-free graphs. *Algorithmica*, 82(8):2360–2394, 2020. doi:10.1007/S00453-020-00730-6.

15 Timothy M Chan. Speeding up the four russians algorithm by about one more logarithmic factor. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 212–217. SIAM, 2014.

16 Timothy M Chan. Finding triangles and other small subgraphs in geometric intersection graphs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1777–1805. SIAM, 2023. doi:10.1137/1.9781611977554.CH68.

17 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985. doi:10.1137/0214017.

18 Fan R. K. Chung, Ronald L. Graham, and Richard M. Wilson. Quasi-random graphs. *Combinatorica*, 9(4):345–362, 1989. doi:10.1007/BF02125347.

19 Mina Dalirrooyfard, Surya Mathialagan, Virginia Vassilevska Williams, and Yinzhan Xu. Towards optimal output-sensitive clique listing or: Listing cliques from smaller cliques. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 923–934, 2024. doi:10.1145/3618260.3649663.

20 Mina Dalirrooyfard and Virginia Vassilevska Williams. Induced cycles and paths are harder than you think. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 531–542. IEEE, 2022. doi:10.1109/FOCS54457.2022.00057.

21 Peter Gartland, Daniel Lokshtanov, Tomáš Masařík, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. Maximum weight independent set in graphs with no long claws in quasi-polynomial time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 683–691, 2024.

22 Dániel Gerbner and Cory Palmer. Survey of generalized tur\'an problems–counting subgraphs. *arXiv preprint arXiv:2506.03418*, 2025.

23 Jan Goedgebeur, Jorik Jooken, Karolina Okrasa, Paweł Rzążewski, and Oliver Schaudt. Minimal obstructions to c_5-coloring in hereditary graph classes. *arXiv preprint arXiv:2404.11704*, 2024.

24 Petr A Golovach, Daniël Paulusma, and Bernard Ries. Coloring graphs characterized by a forbidden subgraph. In *International Symposium on Mathematical Foundations of Computer Science*, pages 443–454. Springer, 2012. doi:10.1007/978-3-642-32589-2_40.

25 Ervin Győri and Hao Li. The maximum number of triangles in c2k+ 1-free graphs. *Combinatorics, Probability and Computing*, 21(1-2):187–191, 2012. doi:10.1017/S0963548311000629.

26 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 1–10, 1977.

27 Tao Jiang and Liana Yepremyan. Supersaturation of even linear cycles in linear hypergraphs. *Combinatorics, Probability and Computing*, 29(5):698–721, 2020. doi:10.1017/S0963548320000206.

28 Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 405–418, 2023. doi:10.1145/3564246.3585157.

29 Matthew Johnson, Barnaby Martin, Jelle J Oostveen, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. Complexity framework for forbidden subgraphs i: The framework. *Algorithmica*, 87(3):429–464, 2025. doi:10.1007/S00453-024-01289-2.

30 Marcin Kamiński. Max-cut and containment relations in graphs. *Theoretical Computer Science*, 438:89–95, 2012. doi:10.1016/J.TCS.2012.02.036.

31 Haim Kaplan, Katharina Klost, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Triangles and girth in disk graphs and transmission graphs. *arXiv preprint arXiv:1907.01980*, 2019. arXiv:1907.01980.

32 Alexandr Kostochka, Dhruv Mubayi, and Jacques Verstraëte. Turán problems and shadows iii: expansions of graphs. *SIAM Journal on Discrete Mathematics*, 29(2):868–876, 2015. doi:10.1137/140977138.

33 Tamás Kovári, Vera Sós, and Pál Turán. On a problem of k. zarankiewicz. In *Colloquium Mathematicum*, volume 3, pages 50–57. Polska Akademia Nauk. Instytut Matematyczny PAN, 1954.

34 Mária Šurimová, Borut Lužar, and Tomáš Madaras. Adynamic coloring of graphs. *Discrete Applied Mathematics*, 284:224–233, 2020. doi:10.1016/J.DAM.2020.03.038.

35 Terence Tao. Structure and randomness in combinatorics. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 3–15. IEEE, 2007. doi:10.1109/FOCS.2007.17.

36 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

37 Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 455–464, 2009. doi:10.1145/1536414.1536477.

38 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654. IEEE, 2010. doi:10.1109/FOCS.2010.67.

39 Huacheng Yu. An improved combinatorial algorithm for boolean matrix multiplication. *Information and Computation*, 261:240–247, 2018. doi:10.1016/J.IC.2018.02.006.

40 Yufei Zhao. *Graph Theory and Additive Combinatorics: Exploring Structure and Randomness*. Cambridge University Press, 2023.