Electronic Colloquium on Computational Complexity, Report No. 34 (2025)

Range Avoidance in Boolean Circuits via Turan-type Bounds

Neha Kuntewar* Jayalal Sarma*

Abstract

Given a circuit $C : \{0,1\}^n \to \{0,1\}^m$ from a circuit class $\mathcal{C}$, with $m > n$, finding a $y \in \{0,1\}^m$ such that $\forall x \in \{0,1\}^n$, $C(x) \neq y$, is the *range avoidance problem* (denoted by $\mathcal{C}$-AVOID). Deterministic polynomial time algorithms (even with access to NP oracles) solving this problem are known to imply explicit constructions of various pseudorandom objects like hard Boolean functions, linear codes, PRGs etc.

Deterministic polynomial time algorithms are known for NC^0_2-AVOID when $m > n$, and for NC^0_3-AVOID when $m \geq \frac{n^2}{\log n}$, where NC^0_k is the class of circuits with bounded fan-in which have constant depth and the output depends on at most k of the input bits. On the other hand, it is also known that NC^0_3-AVOID when $m = n + O\left(n^{2/3}\right)$ is at least as hard as explicit construction of rigid matrices. In fact, algorithms for solving range avoidance for even NC^0_4 circuits imply new circuit lower bounds.

In this paper, we propose a new approach to solving range avoidance problem via hypergraphs. We formulate the problem in terms of Turan-type problems in hypergraphs of the following kind - for a fixed k-uniform hypergraph $\mathcal{H}'$, what is the maximum number of edges that can exist in a k-uniform hypergraph $\mathcal{H}$ which does not have a sub-hypergraph isomorphic to $\mathcal{H}'$? We first demonstrate the applicability of this approach by showing alternate proofs of some of the known results for range avoidance problem using this framework.

We then use our approach to show (using several different hypergraph structures for which Turan-type bounds are known in the literature) that there is a constant c such that MONOTONE-NC^0_3-AVOID can be solved in deterministic polynomial time when $m > cn^2$. To improve the stretch constraint to linear, we show a new Turan-type theorem for a hypergraph structure (which we call the *loose $\mathcal{X}_{2\ell}$-cycles*). More specifically, we prove that Any connected 3-uniform linear hypergraph with $m > n$ edges must contain a loose $\mathcal{X}_{2\ell}$ cycle. Using this, we show that MONOTONE-NC^0_3-AVOID can be solved in deterministic polynomial time when $m > n$, thus improving the known bounds of NC^0_3-AVOIDfor the case of monotone circuits. In contrast, we note that efficient algorithms for solving MONOTONE-NC^0_6-AVOID, already imply explicit constructions for rigid matrices.

Building on this further, we show that there is a polynomial time algorithm for SYMMETRIC-NC^0_3-AVOID when $m > 8n$. We also generalize our argument to solve the special case of range avoidance for NC^0_k where each output function computed by the circuit, is the majority function on its inputs where $m > n^2$.

*Indian Institute of Technology Madras, Chennai, India.
Email: neha.kuntewar@gmail.com, jayalal@cse.iitm.ac.in

ISSN 1433-8092

Contents

1 Introduction

Let $C : \{0,1\}^n \to \{0,1\}^m$ be a Boolean circuit with $\{\wedge, \vee, \neg\}$ gates, with $m > n$. The range of the function represented by the circuit : $\mathsf{Range}(C) = \{C(x) \mid x \in \{0,1\}^n\}$. Clearly, $\exists y \in \{0,1\}^m$ such that $y \notin \mathsf{Range}(C)$. The range avoidance problem (denoted by AVOID) asks, given a circuit C, with $m > n$, find a $y \notin \mathsf{Range}(C)$.

The AVOID problem (introduced by [KKMP21]) has been shown to have connections to some of the central research questions in circuit lower bounds and pseudorandomness. In particular, even $\mathsf{FP}^{\mathsf{NP}}$ algorithm for AVOID is known to imply new circuit lower bounds [Kor22] and new constructions of many other pseudorandom objects [Kor22].

On the algorithms side, AVOID has a trivial $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm. Indeed, given a circuit $C : \{0,1\}^n \to \{0,1\}^m$ with $m > n$, choose $y \in \{0,1\}^m$ and use the NP oracle to check if $\exists x \in \{0,1\}^n$ such that $C(x) = y$. Since $m > n$, there are at least $\frac{1}{2}$ fraction of ys which are outside $\mathsf{Range}(C)$, and hence the algorithm succeeds with at least $\frac{1}{2}$ probability. Designing a deterministic polynomial time algorithm, with access to an NP oracle ($\mathsf{FP}^{\mathsf{NP}}$ algorithm) to solve AVOID is a central open problem. Recently, [CHR] obtained the first single-valued $\mathsf{FS_2P}$ algorithm for AVOID which works infinitely often. Subsequently, [Li24] gave an improved $\mathsf{FS_2P}$ algorithm that works for all n, thus establishing explicit functions in $\mathsf{S_2E}$ requiring maximum circuit complexity. [CHR] showed an unconditional zero-error pseudodeterministic algorithm with an NP oracle and one bit of advice that solves AVOID for infinitely many inputs. These results imply pseudo-determinsitic constructions for Ramsey graphs, rigid matrices, pseudo-random generators etc.(See [CHR]). [ILW23] shows that if there is a deterministic polynomial time algorithm for AVOID then either $\mathsf{NP} = \mathsf{coNP}$ or there does not exist JLS secure iO.

Given the central nature of the problem, it is also meaningful to consider simpler versions first : consider $\mathcal{C}$-AVOID to be the restricted version of AVOID where the circuit is guaranteed to be from the class $\mathcal{C}$. *For which classes of circuits $\mathcal{C}$ do we have an efficient (or $\mathsf{FP}^{\mathsf{NP}}$) algorithm for $\mathcal{C}$-AVOID?* On this frontier, Ren, Santhanam and Wang [RSW22] showed that an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for $\mathcal{C}$-AVOID implies breakthrough lower bounds even when $\mathcal{C}$ restricted to weaker circuit models such as AC^0 and NC^1 circuits. To go down even further, for every constant k, consider the restricted class of circuits NC^0_k where the depth of the circuit is constant and each output bit depends on at most k of the input bits. In a surprising result, Ren, Santhanam and Wang [RSW22] showed that an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for NC^0_4-AVOID implies $\mathsf{FP}^{\mathsf{NP}}$ algorithms for NC^1-AVOID. Additionally, this will also imply new circuit lower bounds - that there is a family of functions in E^{NP} that requires circuits of depth at least $\Omega(n^{1-\epsilon})$.

Complementing the above, [RSW22] also exhibited $\mathsf{FP}^{\mathsf{NP}}$ algorithm for AVOID, when $\mathcal{C}$ is restricted to De Morgan formulas of size s with $m > n^{\omega(\sqrt{s}\log s)}$. At the low-end regime, [GLW22] showed a polynomial time algorithm for NC^0_2 class, where each output bit depends on at most 2 input bits. They show a general template for obtaining $\mathsf{FP}^{\mathsf{NP}}$ algorithms for restricted classes via hitting set constructions. In particular, they give $\mathsf{FP}^{\mathsf{NP}}$ algorithms for NC^0_k circuits, de Morgan formulas, CNF or DNF, provided the stretch is large enough. En route they also give a general method for obtaining the hitting set in polynomial time using approximation degree of polynomials. Complementing this further, [GGNS23] designed deterministic polynomial time algorithms for all NC^0_k-AVOID for $m \geq \frac{n^{k-1}}{\log n}$. For $k = 3$, which was the frontier beyond [GLW22], this requires $m \geq \frac{n^2}{\log n}$. They also showed the reason for lack of progress for NC^0_3-AVOID by proving that a deterministic polynomial time algorithm for NC^0_3-AVOID , where $m = n + O(n^{2/3})$ would imply explicit construction of rigid matrices in deterministic polynomial time. This demonstrates the importance of the stretch function in the context of AVOID problem.

Our Results: In this paper, we propose a new approach to range avoidance problem for NC^0_k circuits via Turan-type extremal problems in hypergraphs. For an NC^0_k circuit C, for each function $f_i \in C$ where $i \in [m]$, let $I(f_i)$ denote the set of input variables that f_i depends on. Let $\mathcal{H}_C$ denote the hypergraph defined as follows: Let V be the set of inputs in C. For $1 \leq i \leq m$, define a hyperedge e_i as $\{x_j \mid j \in [n], x_j \in I(f_i)\}$. Thus $E = \{\{e_i \mid i \in [m]\}\}$ has exactly m edges, each of size at most k. Let $\mathcal{F}$ be the family of functions that appear in the circuit C, and let $\phi : E \to \mathcal{F}$ be a labelling of the edges of the hypergraph $\mathcal{H}$ with the corresponding function. Without loss of generality, by assigning colors to Boolean bits, say red R for 0, and blue B for 1, we can interpret these functions as functions from $\{R, B\}^k \to \{R, B\}$, which induces a color to the hyperedge given any 2-coloring of the vertices of the hypergraph. A 2-coloring of the hyperedges of $\mathcal{H}$, is said to be a *ϕ-coloring* if there is a vertex coloring which induces this edge coloring via ϕ. With this notation, we state the following theorem, which essentially follows from the above definition (see section 3).

Theorem 1.1. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a circuit and $\mathcal{H}_C$ be the corresponding hypergraph and let ϕ be the labelling function. Suppose $\mathcal{H}_C$ contains a sub-hypergraph $\mathcal{H}'$ (of fixed size) such that there is an edge-coloring of $\mathcal{H}'$ which is not ϕ-coloring. Then there is a polynomial time algorithm to find a string outside the range of C.*

The main idea of the above proof is that it suffices to identify an sub-hypergraph $\mathcal{H}'$ in $\mathcal{H}$ such that there is an edge-coloring of $\mathcal{H}'$ which is not ϕ-coloring on $\mathcal{H}'$. If we can ascertain the existence of a copy of $\mathcal{H}'$ by extremal hypergraph theory, then it can be used to solve the AVOID problem. In particular, this formulates range avoidance problem in terms of Turan-type extremal problems in hypergraphs of the following kind - *for a fixed family of hypergraphs $\mathcal{H}'$, what is the maximum number of edges that can exist in any hypergraph that avoids any member of the family $\mathcal{H}'$ as an (induced) subgraph*? This is particularly well-studied for k-uniform hypergraph and is denoted by $\mathsf{ex}_k(n, \mathcal{H}')$. Notice that the family $\mathcal{H}'$ may critically depend on the set of functions $\mathcal{F}$, and the mapping ϕ, and hence on the circuit C. A natural question is, are there fixed $\mathcal{H}'$ that works for all circuits in the circuit class[1] for which we are interested to solve AVOID problem.

We first demonstrate immediate simple applications of this framework for designing algorithms for AVOID in restricted settings. As mentioned above, [GLW22] designed a simple iterative algorithm for solving AVOID when the input is restricted to circuits where each output function depends on at most 2 input bits. We show that the same special case can also be solved using our approach. Thus, as a warm up, we derive an alternative proof of the following theorem (originally due to [GLW22]) using our framework.

As second demonstration of our framework, we use tools from extremal graph theory to provide deterministic polynomial time algorithms for AVOID when $\mathcal{F}$ contains only $\wedge$ and $\vee$ functions that depend on exactly k input bits. We remark that such powerful tools are not needed to solve this special case as the iterative algorithmic idea due to [GLW22] can be extended to this case as well. Nevertheless, we argue that it serves as a demonstration of our technique itself. We state both of these applications as the following proposition.

Proposition 1.2. *There is a deterministic polynomial time algorithm for NC^0_2-AVOID when $m > n$. Using the same framework, there is a polynomial time algorithm for solving $\{\text{AND}_k, \text{OR}_k\}$-AVOID, for a fixed k.*

We now demonstrate the main application of this framework. Towards describing the setting of our application, we study the complexity of AVOID in the restricted case of when the circuit is monotone. A first observation is that by applying DeMorgan's law we can reduce the AVOID (when $m > 2n$) in polynomial time to $\mathcal{C}$-AVOID where $\mathcal{C}$ is restricted to monotone circuits, where reduction preserves the depth of the circuit and at most doubles the size of the circuit. Hence solving AVOID even for monotone circuits.

Proposition 1.3. *If $m > 2n$,* AVOID *reduces to* MONOTONE-AVOID *in polynomial time.*

Proof. Let $C : \{0,1\}^n \to \{0,1\}^m$ a multi-output circuit with $m > 2n$ which is an instance of AVOID. We describe how to obtain the circuit $C' : \{0,1\}^{2n} \to \{0,1\}^m$ from C. By applying De

[1] We remark that while it may not be easy to find the hypergraph structure from the given circuit in general, for the important special cases of the problem mentioned in the previous discussion, just the exhaustive search based on the circuit structure will yield efficient algorithms to find the hypergraph structure.

Morgan's law to push down the negation gates (with appropriate duplication of each gate), we can construct a circuit D equivalent to circuit C, with all the negations at the leaves. The circuit D has input literals $\{x_1, \dots x_n, \neg x_1, \dots \neg x_n\}$. Now obtain the circuit C' by replacing each input variable $\neg x_i$ by a new variable x'_i. Observe that $\mathsf{Range}(C) \subseteq \mathsf{Range}(C')$. Indeed, consider an arbitrary $y \in \mathsf{Range}(C)$ with $a = (a_1, \dots a_n) \in \{0,1\}^n$ such that $C(a) = y$. Let $a' = (a_1, \dots a_n, \overline{a_1}, \dots \overline{a_n}) \in \{0,1\}^{2n}$. Observe that $C'(a') = y$. Since $m > 2n$, C' is a valid input instance for MONOTONE-AVOID. Hence, it suffices to solve the range avoidance problem for monotone circuits with $m > 2n$. $\square$

Therefore, we can restrict our attention to solving the range avoidance problem for the monotone circuits. We also observe the following corollary for the case of NC^0_k circuits which are the current boundary for polynomial time algorithms solving AVOID. For any $k > 0$, if $m > 2n$, NC^0_k-AVOID reduces to MONOTONE-NC^0_{2k}-AVOID in deterministic polynomial time. In particular, when $m > 2n$, NC^0_3-AVOID reduces to MONOTONE-NC^0_6-AVOID in deterministic polynomial time. In the remaining part of the paper, we use the framework in theorem 1.1 to show polynomial time algorithms for MONOTONE-NC^0_3-AVOID and related problems.

Deterministic Polynomial Time Algorithm for MONOTONE-NC^0_3-AVOID with Quadratic stretch: We now show the main technical application of the framework in the case when $\mathcal{F}$ contains only MAJ function on k inputs, with the additional constraint that two functions should depend on at most one common input variable. As per the above formulation, this makes the hypergraph to be linear and k-uniform. In the setting of k-uniform linear hypergraphs, using Turan-type results for specifically designed graphs in $\mathcal{F}$, we can use theorem 1.1 for deriving polynomial time algorithms for solving AVOID for specific class of circuits.

Turan-type extremal problems for hypergraphs were introduced by [BES73] and bounds are known (see [Kee11] for a survey) for $\mathsf{ex}_k(n, \mathcal{H}')$ for a few hypergraphs $\mathcal{H}'$. Bounds are known for when $\mathcal{H}'$ is $k \times k$ grid [FR13], wickets [Sol24], fano plane [KS04] (see section 2 for a brief overview of Turan-type extremal problems, and exact definition of these hypergraphs). We exhibit several different fixed hypergraphs $\mathcal{H}'$ and use them to provide different proofs of the following algorithmic upper bound

Theorem 1.4. MONOTONE-NC^0_3-AVOID *when $m > cn^2$ for any constant c can be solved in deterministic polynomial time.*

Our proof relies on a reduction of the problem to a more restricted case of MONOTONE-NC^0_3-AVOID where each individual output function is the majority of three input bits. We call this version as MAJ_3-AVOID. Notice that for any two functions, there can be at most two input bits that both of them can depend on. By using a combinatorial argument on the circuit, we show how to reduce MAJ_3-AVOID to the case where two functions can depend on at most one common input. We denote this version as ONE-INTERSECT-MAJ_3-AVOID. As mentioned above, we use theorem 1.1 and $\mathcal{H}$ being k-uniform, and $\mathcal{F} = \{\text{MAJ}_3\}$ use it to design polynomial time algorithms for ONE-INTERSECT-MAJ_3-AVOID where each function is majority on three inputs and two different functions depend on at most one common input bit. We exhibit several different fixed hypergraphs $\mathcal{H}'$ - wickets (lemma 3.7), k-cage (lemma A.4), weak Fano plane (lemma A.6), 3×3 grid (lemma 3.9), (k, ℓ)-butterfly (lemma A.8), (k, ℓ)-odd kite (lemma A.11) which can be also used to derive polynomial time algorithms for ONE-INTERSECT-MAJ$_3$-AVOID. Extremal bounds are known for $\mathsf{ex}_k(n, F)$ only when F is weak Fano plane, 3×3 grid and the wicket. The best known bounds for $\mathsf{ex}_k(n, F)$ among these is when F is the hypergraph called a wicket (see section 2 for a definition), and hence we use it in the proof of theorem 1.4.

Observing that the above reduction to monotone case also doubles the number of input bits on which each function depends, for NC^0_k-AVOID with $m > 2n$, this implies a reduction to MONOTONE-NC^0_{2k}-AVOID. We use theorem 1.1, with $\mathcal{H}$ as the $k \times k$ grid (see (lemma 3.9)), we show that:

Theorem 1.5. *There is a deterministic polynomial time algorithm for* ONE-INTERSECT-MAJ$_k$-AVOID *when $m > n^2$.*

However, unlike the case when $k = 3$, it is unclear how to reduce MONOTONE-NC^0_k-AVOID to MAJ$_k$-AVOID, and then further to ONE-INTERSECT-MAJ$_k$-AVOID. Indeed, designing polynomial time algorithms for MONOTONE-NC^0_6-AVOID itself with $m = n + O(n^{2/3})$ already leads to explicit construction of rigid matrices [GGNS23], which is an important problem in the area.

Deterministic Polynomial Time Algorithm for MONOTONE-NC^0_3-AVOID with Linear Stretch: Deterministic polynomial time algorithms are known [GGNS23] for NC^0_3-AVOID when $m \geq \frac{n^2}{\log n}$ and as mentioned above, improving the stretch constraint to $m = n + O(n^{2/3})$ would imply explicit construction of rigid matrices. We next aim to improve the stretch requirement in the above theorem to linear (in fact, to just n), thus improving the known bounds for the case of MONOTONE-NC^0_3-AVOID.

Towards this, notice that the above argument for theorem 1.4 uses the bounds for Turan number from the literature in a blackbox manner. In fact, the quadratic constraints on the stretch function m that we have imposed in theorem 1.4 can be relaxed by using stricter variants of the *power-bound conjecture* in the context of Turan numbers of linear hypergraphs. In particular, [GL21] conjectures that there exists an ϵ such that any linear hypergraph on n vertices having more than $\omega(n^{2-\epsilon})$ edges must contain a $(u, u-4)$-hypergraph[2] as a subgraph. The specific hypergraph of wicket is an example of a $(9, 5)$-hypergraph. However, there are $(9, 5)$-hypergraphs which are not suitable for our purpose (See appendix A.3). Hence, we need a stronger variant of this conjecture which insists on having wicket as a subgraph instead of just $(9, 5)$-subgraphs.

Specifically, in the case of wickets, which we critically use in our argument for theorem 1.4, [FS24, Sol24] conjectured that the Turan number for 3-uniform hypergraphs avoiding wickets is $n^{2-o(1)}$, and this will directly improve the constraint on m as $m = n^{2-o(1)}$ for theorem 1.4 as well.

To go beyond the limitations posed by the above structures for which Turan number bounds are classically studied, we define a new notion of cycles called $\mathcal{X}_{2\ell}$-cycles in 3-uniform linear hypergraphs. We first show a new Turan-type theorem for such cycles for connected hypergraphs, which might be of independent interest.

[2] A $(u, u-4)$-hypergraph in this context is a 3-uniform linear hypergraph which has $u-4$ edges spanning u vertices.

Theorem 1.6. *Any connected 3-uniform linear hypergraph with $m > n$ edges must contain a loose $\mathcal{X}_{2\ell}$ cycle.*

In the context of ONE-INTERSECT-MAJ$_3$-AVOID where $m > n$, using the above theorem, we show that the corresponding hypergraph $\mathcal{H}_C$ contains a loose $\mathcal{X}_{2\ell}$ cycle. Using the framework of theorem 1.1, where we show (lemma 4.5) there exists an edge-coloring of $\mathcal{X}_{2\ell}$ which is not MAJ-coloring. Inaddition, we note that although the cycle is not of fixed size, we can find it in $\mathcal{H}_C$ in polynomial time. This gives us the following theorem.

Theorem 1.7. *(**Main Theorem**) For $m > n$,* MONOTONE-NC_3^0-AVOID *can be solved in deterministic polynomial time.*

Thus, while the applicability of theorem 1.1 seems to impose constraints such as k-uniformity and one-intersection to the cases of AVOID that they can be used to solve, the above theorem indicates that along with other combinatorial reductions to such special cases, it can be still lead to useful bounds for the more general problem. Building on the above, we can also prove the following theorem for the case of circuits where each output function is restricted to be symmetric.

Theorem 1.8. *For $m > 8n$,* SYMMETRIC-NC_3^0-AVOID *can be solved in deterministic polynomial time.*

We prove this by showing a polynomial time reduction from SYMMETRIC-NC_3^0-AVOID to the MONOTONE-NC_3^0-AVOID when $m > 8n$, which is based on a careful combinatorial analysis of the functions involved.

Related Work: More recently, [KPI25] described polynomial time algorithms for solving NC_k^0-AVOID when the stretch function is $m > \tilde{\Omega}(n^{\frac{k}{2}-\frac{k-2}{2(k+2)}})$ when k is an odd constant, and $m > \Omega(n^{\frac{k}{2}} \log n)$ when k is an even constant. This improves the algorithms in [GGNS23] for NC_k^0. We note that our algorithm (theorem 1.7) is for $k = 3$ and works for linear stretch, but in the case of monotone NC_3^0-AVOID. Thus, our results are incomparable with [KPI25].

2 Preliminaries

We study the range avoidance problem for restricted circuit classes. $\mathcal{C}$-AVOID is the following problem: Given a multi-output circuit $C : \{0,1\}^n \to \{0,1\}^m$ such that $m > n$ where each output function can be computed by a circuit in class $\mathcal{C}$, find a $y \in \{0,1\}^m$ which is outside the range of C. In particular, we study the following two problems: MONOTONE-AVOID and SYMMETRIC-AVOID where the circuit class $\mathcal{C}$ is restricted to monotone and symmetric functions respectively.

Hypergraphs: We collect the preliminaries from theory of hypergraphs that we use in the paper. We will work with hypergraphs $G(V, E)$ where $E \subseteq 2^V$. A hypergraph is linear if two edges intersect at most one vertex - $\forall e_1, e_2 \in E$, $|e_1 \cap e_2| \leq 1$. A hypergraph is said to be k-uniform if every edge has exactly k elements from V - $\forall e \in E, |e| = k$. We will be working with k-uniform linear hypergraphs, and in particular with $k = 3$. We will define some of the hypergraphs that are used in the paper.

A $k \times k$ grid in a hypergraph is a set of k^2 vertices $\{v_{11}, v_{12}, \dots v_{kk}\}$ such that there are edges $r_1, r_2, \dots r_k, c_1, c_2, \dots c_k \in E$, corresponding to the vertices in the rows and columns respectively, when the vertices are arranged in the row major order (See fig. 1(a)). A particular special hypergraph (for $k = 3$) is called a *wicket* is the 3×3 grid with one edge removed (See fig. 1(b)).

A Berge path is defined as $v_1 e_1 v_2 \dots v_k e_k v_{k+1}$ where each edge e_i contains vertices v_i, v_{i+1}. A Berge path is said to be even if k is even and odd otherwise. We say a hypergraph is connected if there exists a Berge path between any two pair of vertices.

Turan-type Problems in Hypergraphs: One of the classical extremal Turan-type problems introduced in the context of hypergraphs by [BES73] is the following : *for a fixed set of k-uniform hypergraphs $\mathcal{F}$, what is the maximum number of edges that a k-uniform linear hypergraph can have, if it does not contain a subgraph isomorphic to any of the hypergraphs in the collection $\mathcal{F}$ of hypergraphs.* This number is denoted by $\mathsf{ex}_k(n, \mathcal{F})$. The Turan density of $\mathcal{F}$ is denoted by $\pi(\mathcal{F}) = \lim_{n\to\infty} \left(\binom{n}{k}^{-1} \mathsf{ex}_k(n, \mathcal{F})\right)$. It is known that a k-uniform hypergraph $\mathcal{F}$ has $\pi(\mathcal{F}) = 0$ (also called *degenerate*) if and only if it is k-partite.

The special 3-uniform hypergraph called *wicket*, denoted by W (see figure 1(b)) forms an important step in our argument. We will need the following result about 3-uniform linear hypergraphs due to Solymosi [Sol24].

Proposition 2.1 ([Sol24]). *If a 3-uniform linear hypergraph does not contain a wicket then the number of hyperedges is bounded by $o(n^2)$.*

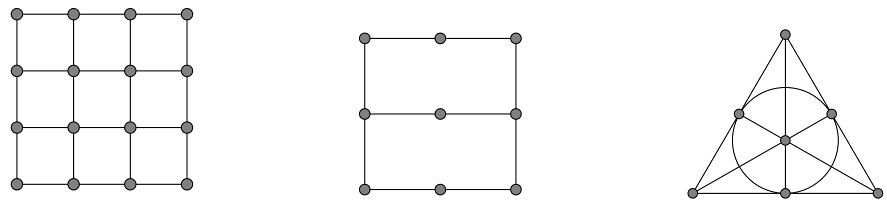
A slightly weaker result was proven by [GS22] where they showed that $\mathsf{ex}_3(n, W) \le \frac{(1-c)n^2}{6}$. Another standard hypergraph that we will be using is the *Fano plane*. A Fano plane F is a 3-uniform linear hypergarph which is isomorphic to the hypergraph $H(V, E)$ with vertex set $V = [7]$ and edge set $E = \{\{1, 2, 3\}, \{3, 4, 5\}, \{1, 5, 6\}, \{3, 6, 7\}, \{2, 5, 7\}, \{1, 4, 7\}, \{2, 4, 6\}\}$. The following result shows a bound on $\mathsf{ex}_3(n, F)$.

Proposition 2.2 ([KS04]). *If a 3-uniform linear hypergraph contains more than $\binom{n}{3} - \binom{\lfloor n/2 \rfloor}{3} - \binom{\lceil n/2 \rceil}{3}$ then it must contain a Fano plane F as a sub-hypergraph.*

Coming to k-uniform hypergraphs, we use the following result about $k \times k$ grid G_k, which exhibits a bound for $\mathsf{ex}_k(n, \{G_k\})$.

Proposition 2.3 ([FR13]). *Let $\mathcal{H}$ be a k-uniform linear hypergraph with $m > \frac{n(n-1)}{k(k-1)}$ edges. Then $\mathcal{H}$ contains a $k \times k$ grid G_k as a sub-hypergraph.*

Cycles in Hypergraphs: Various notions of cycles have been explored in the hypergraph setting. [CCGJ18] consider the notion of linear cycles of length ℓ (denoted by C_ℓ) which is defined as set of ℓ edges such that each pair of adjacent edges e_i, e_{i+1} (modulo ℓ) intersect in exactly one vertex and each pair of non-adjacent edges are disjoint. [CCGJ18] show that for $\mathsf{ex}_k(n, C_{2\ell}) \le c_{k,\ell} n^{1+\frac{1}{\ell}}$



(a) (b) (c)

Figure 1: (a) 4×4 grid (b) A wicket (c) Fano plane

and $\mathsf{ex}_k(n, C_{2\ell+1}) \leq c'_{k,\ell} n^{1+\frac{1}{\ell}}$. Another notion of cycles which is well-studied is that of *Berge cycle of length* ℓ, denoted by $\mathcal{B}_\ell$ which consists of ℓ distinct edges such that each hyperedge e_i contains vertices v_i, v_{i+1} where $1 \leq i < \ell$ and the edge e_ℓ contains vertices v_1, v_ℓ where $v_1, \dots v_\ell$ are distinct vertices. [GL12] show that $\mathsf{ex}_k(n, \mathcal{B}_{2\ell}) = d_{k,\ell} n^{1+\frac{1}{\ell}}$ and $\mathsf{ex}_k(n, \mathcal{B}_{2\ell+1}) = d'_{k,\ell} n^{1+\frac{1}{\ell}}$.

$\mathcal{X}_{2\ell}$-cycles and Loose $\mathcal{X}_{2\ell}$-cycles: We define a new notion of cycles called $\mathcal{X}_{2\ell}$-cycles in 3-uniform linear hypergraphs. We define $\mathcal{X}_{2\ell}$ to be a relaxation of $\mathcal{B}_{2\ell}$ where $\exists e_i, e_j$ such that $|e_i \cap e_j| = 1$, $i + 1 \equiv j \mod 2$ and $|i - j| > 2$. We call the (e_i, e_j)-pair a χ-structure in the cycle. Alternatively, we can view $\mathcal{X}_{2\ell}$ using the lens of Berge path. $\mathcal{X}_{2\ell}$ can be thought as $\mathcal{B}_{2\ell_1} \cup \mathcal{B}_{2\ell_2} \cup \chi$ where $\mathcal{B}_{2\ell_1} = u_1 e_1 u_2 \dots u_{2\ell_1} e_{2\ell_1} u_{2\ell_1+1}$, $\mathcal{B}_{2\ell_2} = v_1 e'_1 v_2 \dots v_{2\ell_2} e'_{2\ell_2} v_{2\ell_2+1}$, $\chi = (e, e')$ where $e, e' \notin \{e_1, \dots e_{2\ell_1}, e'_1, \dots e'_{2\ell_2}\}$ and $e = \{u_1, u_{\ell_1}, w\}, e' = \{v_1, v_{\ell_2}, w\}$. If we relax the Berge paths to Berge walks (denoted by $\mathcal{P}_\ell$) by allowing repetition of edges and vertices, we get loose $\mathcal{X}_{2\ell}$ cycle. In section 4, we shall prove extremal bounds on this structure and use it to solve range avoidance for restricted circuit classes.

3 Range Avoidance for NC^0_k via Fixed Hypergraphs

In this section, we describe the main technical tool of the paper by formulating the range avoidance problem as a way of avoiding certain hypergraph - leading to a Turan-type formulation of the problem.

We recall the notation from the introduction: for an NC^0_k circuit $C = (f_i)_{i \in [m]} : \{0,1\}^n \to \{0,1\}^m$. Let $\mathcal{H}_C$ denote the hypergraph defined as follows: The set of vertices are $[n]$. For $1 \leq i \leq m$, define a hyperedge e_i as $\{x_j \mid j \in [n], x_j \in I(f_i)\}$. Thus $E = \{\{e_i \mid i \in [m]\}\}$ has exactly m edges, each of size at most k. Let $\mathcal{F}$ be the family of functions that appear in the circuit C, and let $\phi : E \to \mathcal{F}$ be a labelling of the edges of the hypergraph $\mathcal{H}$ with the corresponding function. A 2-coloring of the hyperedges of $\mathcal{H}$, is said to be a *ϕ-coloring* if there is a vertex coloring which induces this edge coloring via ϕ. We argue the following:

Theorem 1.1. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a circuit and $\mathcal{H}_C$ be the corresponding hypergraph and let ϕ be the labelling function. Suppose $\mathcal{H}_C$ contains a sub-hypergraph $\mathcal{H}'$ (of fixed size) such that there is an edge-coloring of $\mathcal{H}'$ which is not ϕ-coloring. Then there is a polynomial time algorithm to find a string outside the range of C.*

Proof. Let $\mathcal{H}_C$ be the hypergraph corresponding to the circuit C. Let $\mathcal{H}'$ be a sub-hypergraph of fixed size ℓ in $\mathcal{H}_C$. We note that since $\mathcal{H}'$ is of fixed size, we can exhaustively search for a copy of $\mathcal{H}'$ in $\mathcal{H}_C$ in polynomial time. Let $\Gamma : E(\mathcal{H}') \to \{R, B\}$ be an edge-coloring of $\mathcal{H}'$ which in not a ϕ-coloring. Let C' be the circuit corresponding to the hypergraph $\mathcal{H}'$. Let $y = y_1 y_2 \dots y_m$ be defined as follows:

$$y_i = \begin{cases} *, & \text{if } e_i \notin E(\mathcal{H}') \\ 0, & \text{if } e_i \in E(\mathcal{H}') \text{ and } \Gamma(e_i) = R \\ 1, & \text{otherwise} \end{cases}$$

where $*$ denotes that y_i can take any value in $\{0,1\}$. We will show that $y \notin \mathsf{Range}(C)$. Let $\Delta : \{R, B\} \to \{0,1\}$ be a function such that $\Delta(R) = 0, \Delta(B) = 1$. For the sake of brevity, let $\Delta(x) = \Delta(x_1, \dots x_n) = \Delta(x_1)\Delta(x_2)\dots\Delta(x_n)$. It suffices to show the following claim.

Claim 3.1. *$y \in \mathsf{Range}(C)$ if and only if $\Delta(y)$ is a valid ϕ-coloring.*

Proof. Notice that $y \in \mathsf{Range}(C)$ if and only if $\exists x \in \{0,1\}^n$ such that $C(x) = y$. This is equivalent to $\exists \Pi : V \to \{R, B\}$ such that the corresponding $\Gamma : E \to \{R, B\}$ is a $\phi -$ coloring. Indeed, the latter equivalence can be obtained by setting Π, Γ such that $\Pi(v_i) = \Delta(x_i)$ and $\Gamma(e_j) = \Delta(y_j)$ for all $i \in [n], j \in [m]$. □

Since, $\mathcal{H}_C$ has an edge coloring Γ which is not a ϕ-coloring, by claim 3.1 we obtain a $y \in \{0,1\}^m$ which is outside $\mathsf{Range}(C)$. □

Warm-up 1: Algorithm for NC^0_2-Avoid: As mentioned in the introduction, [GLW22] described a polynomial time algorithm for solving the range avoidance problem for NC^0_2 circuit. In the following, we show an alternate proof of the same as an application of the framework described above.

Proposition 3.2. *There is a polynomial time algorithm for NC^0_2-Avoid when $m > n$.*

Proof. Let $C : \{0,1\}^n \to \{0,1\}^m$ be the given NC^0_2 circuit. [GLW22] observed that there are only four types of NC^0_2 functions possible: And, Or, Parity and constant functions. We can check the type of function in polynomial time. Suppose there is a constant function f in the circuit. Wlog let this function be a constant zero function. Then by setting the output of f to 1 and the other output bits arbitrarily we get a string which is outside the range of the circuit.

Now we consider the other case when the circuit contains only $\{\wedge, \vee, \oplus\}$ gates where inputs may be negated. We consider the graph $\mathcal{H}_C$ corresponding to the circuit C. Since $m > n$, the graph $\mathcal{H}_C$ corresponding to the circuit C indeed contains a cycle Q. Furthermore, we can find this cycle in polynomial time. Thus, it suffices to obtain an edge-coloring of Q which is not a ϕ-coloring. We consider the following cases based on the function types of the edges participating in the cycle Q:

Case 1: Suppose each edge in Q computes a Parity of two inputs which may be negated. Consider an edge f with endpoints x_i, x_j. Consider the path P starting at x_i and ending at x_j obtained by deleting f from Q. Let Γ color all the edges in $Q - f$ to R.

Let $x_i = b \in \{R, B\}$. Notice that setting the function to R fixes the color of every other vertex to either b or $\bar{b}$ depending on the function. In particular, it fixes $x_j = b' \in \{b, \bar{b}\}$. This fixes the value of the function computed at f, and hence the color of f. Thus, flipping the color of f would produce an edge-coloring which is not achievable. Therefore, the following coloring Γ is a not a ϕ-coloring.

$$\Gamma(e) = \begin{cases} R & \text{if } e \in Q - f \\ B & \text{if } e = f \text{ and } b' = b \\ R & \text{if } e = f \text{ and } b' = \bar{b} \end{cases}$$

Case 2: Suppose there exists an edge e_1 in Q which computes AND or OR of two inputs which may be negated. Let the cycle Q be $e_1 e_2 \dots e_\ell e_1$. Consider the path $e_1 e_2 \dots e_{\ell-1}$ obtained by deleting the edge e_ℓ from Q. Consider the following edge-coloring $\Gamma : Q - e_\ell \to \{R, B\}$

$$\Gamma(e) = \begin{cases} B & \text{if } e \in Q - e_\ell, \phi(e) = \wedge \\ R & \text{if } e \in Q - e_\ell \text{ and } \phi(e) \in \{\vee, \oplus\} \end{cases}$$

Let $e_1 \cap e_\ell = \{x_i\}$ and $e_\ell \cap e_{\ell-1} = \{x_j\}$. We would like to show that by coloring the edges of $Q - e_\ell$ according to Γ, we end up fixing the colors of x_i, x_j thus fixing the color of e_ℓ or we obtain an inconsistency. In the former case, flipping the color of e_ℓ we obtain an edge-coloring which is not a ϕ-coloring.

It suffices to show that Γ fixes the color of e_ℓ or we obtain an inconsistent coloring for $Q - e_\ell$. We prove this by induction on length of the path. For the base case, wlog let $\phi(e_1) =$AND. According to our Γ, we color e_1 to B, which fixes the color of both vertices incident on e_1. By induction hypothesis, let Γ fix the colors of all vertices participating in the path $e_1 e_2 \dots e_{k-1}$ for $k \in [\ell]$. In the other case when the coloring is inconsistent we are already done. Thus, we would like to argue that Γ fixes the other endpoint y of e_k where $y \notin e_{k-1}$. Notice that, since one input feeding into e_k is already fixed Γ gives a way to fix the other input y to $b \in \{R, B\}$ by setting the color of e_k appropriately or we get an inconsistency at this stage. Thus, inductively either we get an inconsistent coloring (in this case color e_ℓ arbitrarily) or we end up fixing the colors of x_i, x_j which in turn fixes the color of the edge e_ℓ.

Hence, flipping the color of ℓ fixed by the above assignment produces an edge-coloring which is not a ϕ-coloring.

By the above case analysis we have an edge-coloring of $\mathcal{H}'$ which is not ϕ-coloring. By theorem 1.1 we obtain a string outside the range of the circuit in polynomial time. □

Warm-up 2: Polynomial time algorithm for $\{$AND, OR$\}$**-AVOID:** To demonstrate the method further, as another simple application, we show a polynomial time algorithm for solving the range avoidance problem when each output function computes an AND or OR function of k input bits for a fixed $k > 0$ and satisfies the property that any two output functions share at most one input.

As our next application, we show a polynomial time algorithm for solving the range avoidance problem when each output function computes an AND or OR function of k input bits for a fixed $k > 0$. Additionally, it satisfies the constraint that any two output functions share at most one input. Thus, the hypergraph $\mathcal{H}_C$ corresponding to C is a k-uniform linear hypergraph. The sub-hypergraphs we will use in this case are k-crown and C^*. As our next application, we show a polynomial time algorithm for solving the range avoidance problem when each output function computes an AND or OR function of k input bits for a fixed k. The sub-hypergraphs we will use in this case are k-crown and C^*. A *k-crown* is a k-uniform linear hypergraph consisting of k disjoint hyperedges $\{e_1, \ldots e_k\}$ and an additional hyperedge e_0 intersecting each hyperedge $e_1, \ldots e_k$ exactly once. C^* is a variant of the k-crown where the edges $e_1, \ldots e_{k-2}$ intersect in a common vertex $v \notin e_0$. We will use the following extremal bound on these subhypergraphs

Proposition 3.3 ([ZBW24]). *Let $\mathcal{H}$ be a k-uniform linear hypergraph with $m > \frac{k(k-2)(n-s)}{k-1}$ where s is the number of vertices with degree at least $(k-1)^2 + 2$. Then $\mathcal{H}$ contains a k-crown or C^*.*

Proposition 3.4. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a multi-output circuit where $m > \frac{k(k-2)(n-s)}{k-1}$ and each output function computes AND_k or OR_k for fixed k. Additionally, any two output functions share at most one input. Then, there is a polynomial time algorithm for solving AVOID on C.*

Proof. We will show an edge coloring of the hypergraphs k-crown or C^* which is not a ϕ-coloring.

$$\Gamma(e) = \begin{cases} B \text{ if } e \in \{e_1, \ldots e_k\}, \phi(e) = \text{AND} \\ R \text{ if } e \in \{e_1, \ldots e_k\}, \phi(e) = \text{OR} \\ B \text{ if } e = e_0, \exists i \in [k] \text{ such that } \phi(e_i) = \text{OR} \\ R \text{ otherwise} \end{cases}$$

By Proposition 3.3 we have that if $m > \frac{k(k-1)(n-s)}{k-1}$ then $\mathcal{H}$ contains a C^* or k-crown. Furthermore, we can find these subhypergraphs in polynomial time. By the above argument, we can find an edge coloring of C^* and k-crown which is not a ϕ-coloring. By theorem 1.1 we can find a string outside the range of C in polynomial time. ☐

We remark that the above special case of AVOID ({AND, OR}-AVOID) can be solved by a direct algorithm similar to the algorithm for NC_0^2-AVOID due to [GLW22]. Nevertheless the above method is yet another demonstration of our framework for solving AVOID problem.

3.1 Polynomial time algorithm for ONE-INTERSECT-MAJ$_3$-AVOID

In this subsection, we show the first application of our formulation of the problem in terms of hypergraphs. We apply it to describe a deterministic polynomial time algorithm for ONE-INTERSECT-MAJ$_3$-AVOID.

For our purpose, we are interested in a special case of ϕ-coloring when $\mathcal{F} = \{\text{MAJ}\}$. We define this formally below:

Definition 3.5 (MAJ**-coloring).** Let $H(V, E)$ be a k-uniform hypergraph. We say $\Gamma : E \to \{R, B\}$ is a MAJ-coloring if there exists a vertex coloring $\Pi : V \to \{R, B\}$ such that $\forall e \in E(H)$ we have $\Gamma(e) = B \iff \exists S \subseteq e, |S| \geq \left\lfloor \frac{|V|}{2} \right\rfloor + 1$ such that $\forall v \in S,\ \Pi(v) = B$.

We have the following corollary from theorem 1.1.

Corollary 3.6. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a circuit where each output function computes a* MAJ *function. Let $\mathcal{H}_C$ be the corresponding hypergraph. Suppose $\mathcal{H}_C$ contains a subhypergraph $\mathcal{H}'$ such that there is an edge-coloring of $\mathcal{H}'$ which is not* MAJ*-coloring. Then there is a polynomial time algorithm to find a string outside the range of C.*

Thus it is sufficient to exhibit an explicit fixed size linear 3-uniform hypergraph $\mathcal{H}'$ which satisfies the conditions of Corollary 3.6.

Hypergraph Structure : Wickets - A *wicket* is defined as a 3×3 grid with one edge removed. By proposition 2.1, if $\mathcal{H}$ contains more that $o(n^2)$ edges, then it contains a wicket. We will show that there is an edge coloring of a wicket which is not a MAJ-coloring.

Lemma 3.7. *Let $\mathcal{H}$ be a wicket. There exists an edge-coloring $\Gamma : E \to \{R, B\}$ such that it not a* MAJ*-coloring. Furthermore, we can find this in polynomial time.*

Proof. Wlog let the edge e removed from the wicket be a column edge. Let $E = E_1 \cup E_2$ where E_1 be the set of three row edges and E_2 denote the set of column edges. We will show that the following $\Gamma : E \to \{R, B\}$ is not a MAJ-coloring: $\Gamma(e) = \begin{cases} R & \text{if } e \in E_1 \\ B & \text{if } e \in E_2 \end{cases}$.

Let U be the set of vertices upon which the edges of E_2 are incident. For the edges in E_1 to be colored R, there must be at least three vertices in U that should be colored R. Similarly, for the edges in E_2 to be B, there must be at least 4 vertices in U that should be colored B. Thus, totally there should be at least 7 distinct vertices in U. But $|U| = 6$, which is a contradiction. Hence, Γ is not a MAJ-coloring. ☐

The following is an easy corollary of corollary 3.6 and lemma 3.7.

Corollary 3.8. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be an instance of* One-Intersect-Maj$_3$-Avoid *with $m = \Omega(n^2)$. Then we can find a $y \in \{0,1\}^m$ outside the range of C in polynomial time.*

Proof. Let H_C be the 3-uniform linear hypergraph corresponding to the the circuit C. Since $m > o(n^2)$, proposition 2.1 guarantees existence of a wicket in $\mathcal{H}$. Furthermore, we can find such wicket in polynomial time by simply checking across all subsets of m which are of size 5. The number of subsets is bounded by $\binom{m}{5}$. By corollary 3.6 and lemma 3.7 it follows that we can find a y outside the range of C in polynomial time. ☐

3.2 Polynomial time Algorithm for ONE-INTERSECT-MAJ$_k$-AVOID

In this subsection, we show a second application of the framework proposed in theorem 1.1. Let $C : \{0,1\}^n \to \{0,1\}^m$ be a multi-output circuit such that each output function computes MAJ of k input bits. Let $\mathcal{H}$ be a k-uniform hypergraph obtained from C as in the case of theorem 1.1. Since every pair of output functions intersect in at most one input variable, we have that $\mathcal{H}$ is a linear k-uniform hypergraph.

Recall proposition 2.3 which argued that a k-uniform hypergraph that has more than $\frac{n(n-1)}{k(k-1)}$ must have a $k \times k$ grid contained in it. We will show that if $\mathcal{H}$ contains a $k \times k$- grid then we can solve range avoidance for the corresponding circuit in polynomial time and hence this will imply an algorithm for ONE-INTERSECT-MAJ$_k$-AVOID by using the same framework as theorem 1.1.

Lemma 3.9. *There exists an edge-coloring $\Gamma : E \to \{R, B\}$ of $k \times k$-grid which is not a* MAJ*-coloring. Furthermore, we can find Γ in polynomial time.*

Proof. Let $\mathcal{H}(V,E)$ be a $k \times k$ grid. Let $E = E_1 \cup E_2$ where E_1, E_2 are the set of edges corresponding to the rows and columns of the grid respectively. We define an edge-coloring

$$\Gamma(e) = \begin{cases} R & \text{if } e \in E_1 \\ B & \text{otherwise} \end{cases}$$

We will show that Γ is not a valid MAJ-coloring. We consider the following cases based on the parity of k:

Case 1 Suppose k is odd. For each $e \in E_1$ to be colored R, at least $\lfloor \frac{k}{2} \rfloor + 1$ of its vertices should be colored R. Since the edges in E_1n are pairwise disjoint, at least $k\left(\lfloor \frac{k}{2} \rfloor + 1\right)$ distinct vertices should be colored R. Similarly for edges in E_2, we have that at least $k\left(\lfloor \frac{k}{2} \rfloor + 1\right)$ vertices in V should be colored B. Hence, there should be at least $2k\left(\lfloor \frac{k}{2} \rfloor + 1\right) > k^2$ vertices in V, which is a contradiction.

Case 2 Suppose k is even. By previous argument, for edges in E_2 to get color B at least $k\left(\frac{k}{2} + 1\right)$ vertices should be colored B. For edges in E_1 at least $k\left(\frac{k}{2}\right)$ vertices should be colored R. So totally, there are at least $k\,(k+1) > k^2$ vertices in V, which is again a contradiction.

In either case, we obtain a contradiction. Hence, Γ is not a MAJ-coloring. ☐

By proposition 2.3 and lemma 3.2, we have the following result:

Theorem 3.10. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be an instance of* ONE-INTERSECT-MAJ$_k$-AVOID *and $m > \frac{n(n-1)}{k(k-1)}$. There is a polynomial time algorithm to find a string outside the range of C.*

4 Polynomial time Algorithm for MONOTONE-NC$^0_3$-AVOID

In this section, we describe a deterministic polynomial time algorithm for MONOTONE-NC$^0_3$-AVOID for $m = \Omega(n^2)$. The algorithm proceeds in three steps. In the first step, we give a polynomial time reduction from MONOTONE-NC$^0_3$-AVOID to MAJ$_3$-AVOID. Next, we show a reduction from

MAJ_3-AVOID to ONE-INTERSECT-MAJ_3-AVOID. Finally, we describe a polynomial time algorithm for solving MAJ_3-AVOID when $m = \Omega(n^2)$.

4.1 Reduction from MONOTONE-NC_3^0-AVOID to MAJ_3-AVOID:

We show the following reduction:

Theorem 4.1. *There is a polynomial time reduction from* MONOTONE-NC_3^0-AVOID *to* MAJ_3-AVOID.

Proof. Let $C : \{0,1\}^n \to \{0,1\}^m$ be a monotone circuit with $m > n$. We will obtain a circuit $C' : \{0,1\}^{n'} \to \{0,1\}^{m'}$ such that $n' \leq n < m' \leq m$ and each output function computes majority of three input bits. Furthermore, we can find a $y \notin \mathsf{Range}(C)$ from $y' \notin \mathsf{Range}(C')$ in polynomial time by setting the remaining $m - m'$ many bits of y to arbitrary values. We remark that this proof technique is inspired by [GLW22].

To begin with, note that there are only a few types of monotone functions that depend on 3 bits. For each of these cases we will show a reduction from $C : \{0,1\}^n \to \{0,1\}^m$ to a smaller circuit $C' : \{0,1\}^{n-1} \to \{0,1\}^{m-1}$ such that a string outside the range of C' gives a string which is outside the range of C. Let C_j denote the sub-circuit of C that computes the j-th bit of the output. Let $W_i := \{x \in \{0,1\}^3 \mid |x|_1 = i\}$ be the set of inputs with weight i, for $i \in \{0,1,2,3\}$. We describe the reduction for each type of function except when the function is the MAJ_3 function. We iteratively apply this sequence of reduction rules for each $j \in \{1, 2 \ldots m\}$, we will end up a circuit where each output bit is the MAJ_3 function in terms of input variables.

Case 1: $C_j^{-1}(0) = \emptyset$**:** That is C_j computes a constant 1 function. Then setting $y_1 = 0$ and the remaining output bits arbitrarily gives a string outside the range of C.

Case 2: $C_j^{-1}(0) = W_0$**:** Suppose C_j computes OR of three input variables x_1, x_2, x_3. Then setting $y_1 = 0$ and all the input variables x_1, x_2, x_3 to 0 we obtain a smaller circuit $C' : \{0,1\}^{n-3} \to \{0,1\}^{m-1}$ such that $y' \notin Range(C')$ then $y = 0y'$ is outside the range of C.

Case 3: $C_j^{-1}(0) = W_0 \cup \{\alpha\}$ where $\alpha \in W_1$: Without loss of generality, we assume $\alpha = 001$. In this case, setting $y_1 = 0$ and the input variables x_1, x_2 to 0 we obtain a smaller circuit $C' : \{0,1\}^{n-2} \to \{0,1\}^{m-1}$. Same argument applies when $\alpha = 100$, and $\alpha = 010$.

Case 4: $C_j^{-1}(0) = W_0 \cup \{\alpha_1, \alpha_2\}$ where $\alpha_1, \alpha_2 \in W_1$: Without loss of generality, let $\alpha_1 = 001$, and $\alpha_2 = 010$. By property of monotone functions, we have that C_j evaluates to 1 on inputs $\{101, 110, 111\}$. Then setting $y_1 = 0$ and $x_1 = 0$, yields a circuit $C' : \{0,1\}^{n-1} \to \{0,1\}^{m-1}$. Same argument applies in the case when $\alpha_1 = 010$, $\alpha_2 = 100$ and also in the case when $\alpha_1 = 001, \alpha_2 = 100$.

Case 5: $C_j^{-1}(0) = W_0 \cup W_1 \cup \{\alpha\}$ where $\alpha \in W_2$: Without loss of generality, assume $\alpha = 011$. Setting $y_1 = 1$ and $x_1 = 1$ we get a smaller circuit $C' : \{0,1\}^{n-1} \to \{0,1\}^{m-1}$. The other cases when $\alpha = 101$ and $\alpha = 110$ can be handled similarly.

Case 6: $C_j^{-1}(0) = W_0 \cup W_1 \cup \{\alpha_1, \alpha_2\}$ where $\alpha_1, \alpha_2 \in W_2$: Without loss of generality, assume $\alpha_1 = 011, \alpha_2 = 110$. Setting $y_1 = 1$ and $x_1 = 1$ we get a smaller circuit $C' : \{0,1\}^{n-1} \to \{0,1\}^{m-1}$. The same argument applies when $W_2 \setminus \{\alpha_1, \alpha_2\} = 110$ or 011.

Case 7: $C_j^{-1}(0) = W_0 \cup W_1 \cup W_2$**:** In this case, C_j computes AND of three input variables x_1, x_2, x_3. Then setting $y_1 = 1$ and all the input variables x_1, x_2, x_3 to 1 we obtain a smaller circuit $C' : \{0,1\}^{n-3} \to \{0,1\}^{m-1}$ such that $y' \notin Range(C')$ then $y = 1y'$ is outside the range of C.

Case 8: $C_j^{-1}(0) = W_0 \cup W_1 \cup W_2 \cup W_3$**:** That is, C_j computes a constant zero function. Then setting $y_1 = 1$ and the remaining output bits arbitrarily gives a string outside the range of C.

It can be verified that the only function that is not covered in the above cases is when C_j computes MAJ on 3 bits. Note that in each case we eliminate one output bit and at least one input bit. Thus, finally we are left with a circuit $C' : \{0,1\}^{n'} \to \{0,1\}^{m'}$ where $m' > n'$ and each output bit is computed by MAJ_3 function.

We note that we can check the type of function computed by a circuit by simply evaluating the function on all possible input values. Since the function depends on only three variables there are only 8 input possibilities to check. Hence, overall the reduction can be done in polynomial time. □

4.2 Polynomial time Algorithm for MAJ_3-AVOID

We show that there is a deterministic polynomial time algorithm for MAJ_3-AVOID. Recall that MAJ_3 avoid instance is a circuit $C : \{0,1\}^n \to \{0,1\}^m$ with constant depth, bounded fan-in and polynomial size, where each output function of the circuit is a majority of exactly three input variables. For each $i \in [m]$, let f_i denote the function corresponding to the i-th output bit, and let $I(f_i)$ denote the set of three variables that f_i depends on. We shall demonstrate working of the algorithm in three steps.

For our purpose, we shall define sub-circuits of C called *clusters* (denoted by K) as follows: Let $\mathcal{O} = \{f_1, \ldots f_m\}$ be the set of output functions of C. We define the relation $R : \mathcal{O} \times \mathcal{O}$ as follows: We say $(f_i, f_j) \in R$ if $\exists$ some input x that feeds into both f_i, f_j for $i, j \in [m]$. Let R' be transitive closure of R. Observe that R' is an equivalence relation. We define a cluster to be an equivalence class of R'.

Step 1 : Reduction to a single cluster: The following lemma shows that C must contain a cluster with more outputs than inputs and hence we can concentrate on such a cluster.

Lemma 4.2. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a multi-output circuit with $m > n$, then there exists a cluster K such that $|K| > |I(K)|$ where $I(K) = \cup_{f \in K} I(f)$.*

Proof. Let $K_1, \ldots K_t$ be the clusters such that $\cup_{i \in [t]} K_i$ covers the set of output functions of C. Observe that $\cup_{i \in [t]} I(K_i)$ covers the input set of C. By definition, $\forall i, j \in [t]$ such that $i \neq j$ we have that $K_i \cap K_j = \emptyset$ and $I(K_i) \cap I(K_j) = \emptyset$. Assume for the sake of contradiction that for each cluster

K_i, we have $|K_i| \leq |I(K_i)|$. Then by the above observation, we have that $\left|\cup_{i\in[t]} K_i\right| \leq \left|\cup_{i\in[t]} I(K_i)\right|$ implying that $m < n$, which is a contradiction. $\square$

Let K be a cluster guaranteed by proposition 4.2. Let C_K be the sub-circuit of C corresponding to the cluster K. Since, $|K| > |I(K)|$, there must be a string outside the range of C_K. Observe that if $y' \in \{0,1\}^{|K|} \notin \mathsf{Range}(C_K)$ then any $y \in \{0,1\}^m$ which agrees with y' is outside $\mathsf{Range}(C)$. Hence, it suffices to solve the problem on C_K.

It is easy to see that if there exist two output functions that intersect in three inputs then we already have a string outside the range. Hence, we consider the cases when they intersect in fewer than three inputs. Motivated by this, we define the following problems: TWO-INTERSECT-MAJ$_3$-AVOID (ONE-INTERSECT-MAJ$_3$-AVOID) is the following problem: Given $C : \{0,1\}^n \to \{0,1\}^m$ such that output function is MAJ$_3$ function and any two functions share at most two (resp. one) inputs, the goal is to find $y \notin \mathsf{Range}(C)$.

Step 2 : A reduction to ONE-INTERSECT-MAJ$_3$**-**AVOID For the rest of the proof $C_K = C$, $m' = m$ and $n' = n$. We show that the there is polynomial time reduction from TWO-INTERSECT-MAJ$_3$-AVOID problem to ONE-INTERSECT-MAJ$_3$-AVOID problem. Finally, in step 3 we will show that that there is a deterministic polynomial time algorithm for ONE-INTERSECT-MAJ$_3$-AVOID problem.

Lemma 4.3. *There is a polynomial time reduction from* TWO-INTERSECT-MAJ$_3$-AVOID *to* ONE-INTERSECT-MAJ$_3$-AVOID*.*

Proof. Let $C : \{0,1\}^n \to \{0,1\}^m$ be a circuit such that each output bit is computed by a MAJ$_3$ function where $m > n$. By step 1, we know that the set of functions corresponding to C form some cluster K. Hence, we obtain C starting with an arbitrary function in K and iteratively adding a new function f to $\mathcal{O}$, such that $I(f) \cap I(\mathcal{O}) \neq \emptyset$. Our algorithm will follow this construction procedure, and eliminate functions that appear in that order by setting input variables.

We start with the following claim that helps us to eliminate functions from the circuit by setting input variables. Let f_1, f_2 be two output functions of C such that $|I(f_1) \cap I(f_2)| = 2$. Initially $\mathcal{O} = \{f_1, f_2\}$, and we iteratively apply the following claim for each function $g \in K \setminus \mathcal{O}$ that is used to build the cluster by the above process. A variable in $I(K)$ is said to be *alive* if it is not set to a value in $\{0,1\}$ in the below process.

Claim 4.4. *$\forall g \in K \setminus \mathcal{O}$ such that $|I(g) \cap I(\mathcal{O})| = 2$, there is an setting of $g \in \{0,1\}$ and a consistent assignment to some of the input variables, such that the number of variables in $I(\mathcal{O})$ which is alive is at most 1.*

Proof. We shall prove this by induction on the construction of the cluster. Initially, let $\mathcal{O} = \{f_1, f_2\}$, $I(\mathcal{O}) = \{x_1, x_2, x_3, x_4\}$. Without loss of generality, let $I(f_1) = \{x_1, x_2, x_3\}$ and $I(f_2) = \{x_2, x_3, x_4\}$. We observe that if $f_1 = 1$ and $f_2 = 0$, then $x_1 = 1, x_4 = 0, x_3 = \neg x_2$ and the number of variables in $I(\mathcal{O})$ is exactly 1.

By hypothesis, let the number of variables in $I(K)$ which are alive after i steps is at most 1. Now consider a g such that $|I(g) \cap I(K)| = 2$, we would like to show that there is an assignment of values to g (which also forces setting of input variables) such that in the new cluster, $K \cup \{g\}$, the number of variables that are alive is still at most 1. The following cases arise depending on the two inputs g shares with the cluster. Let $z_1, z_2 \in I(g) \cap I(K)$ and let $y = I(g) \setminus I(K), b \in \{0, 1\}$.

Case 1: *Both z_1, z_2 are not alive:* There are two cases to consider. Suppose $z_1 = z_2 = b$. By setting $g = \overline{b}$, we can obtain a string which is outside the range of C. The other case is when $z_1 = b$, $z_2 = \overline{b}$. In this case, we set $g = b$ and $y = b$. Therefore, the number of variables in the new cluster is still at most 1.

Case 2: *Exactly one of z_1, z_2 is alive:* Again two cases arise. Consider the first case when $z_1 = b$, $z_2 = x$. Now, we set $g = x = y = \overline{b}$ eliminating all the variables. The other case is: $z_1 = b$, $z_2 = \neg x$. Similarly, we set $g = \overline{b}$, $x = b$, $y = \overline{b}$ which eliminates all the variables.

Case 3: *Both z_1, z_2 are alive:* If z_1, z_2 that are both alive, then it must be that $z_1 = x, z_2 = \neg x$. In this case, we set $g = y = 0$ so that the number of variables in the cluster is at most 1.

Hence, in all the cases there is an assignment value to g and consistent assignment for variables in $I(g)$ such that there is at most one variable which is alive in $I(\mathcal{O})$. □

Now, we consider the functions h such that $|I(h) \cap I(\mathcal{O})| = 1$. Let $w := I(h) \cap I(\mathcal{O})$. If w is assigned to $b \in \{0, 1\}$, then we set h to $\overline{b}$ and the input variables in $I(h) \setminus \{w\}$ to $\overline{b}$. Consider the other case when w is alive. Wlog let $w = x$. We set the output of h and its other two inputs to $\overline{x}$. In this case we have not yet assigned a fixed $0/1$ value to h, which we shall fix in the next step. However, note that in either case the number of alive variables in $\mathcal{O}$ is at most 1.

After at most $n - 2$ iterations, we would have fixed each of the inputs to one of $0, 1, x$ or $\neg x$. This is because in the iterative process, each function that we add to the set $\mathcal{O}$ covers at least one new variable. Now consider an $f \in K \setminus \mathcal{O}$, which is guaranteed since $m > n$. Observe that $I(f) \subseteq I(\mathcal{O})$. Let $I(f) = \{x_1, x_2, x_3\}$. The following two cases arise based on the number of alive variables.

Case 1: Suppose there is no variable in $\mathcal{O}$ which is alive. Then the value of f is already fixed by the inputs $I(\mathcal{O})$. Wlog let this be $b \in \{0, 1\}$. Setting the output of f to $\overline{b}$ gives a string outside $\mathsf{Range}(C)$.

Case 2: Suppose there exactly on variable that is alive. Wlog let $x_1 = x$.

Case 2(a): Consider the case when $x_2 = x_3 = b$ for $b \in \{0, 1\}$. Notice that this already forces f to take value b. Thus, by setting f to $\overline{b}$ and the functions outputs of functions that were set to x to b we obtain a $y \in \{0, 1\}^m$ which is outside $\mathsf{Range}(C)$.

Case 2(b): Suppose $x_2 = b, x_3 = \overline{b}$. Then setting the output of f to b, we fix the value of x to b. Thus, at this stage there are no variables that are alive in $\mathcal{O}$. Since, $|\mathcal{O}| \leq n - 1$ and $m > n$ there exists a $g \in K \setminus \mathcal{O}$ such that $I(g) \subseteq I(\mathcal{O})$. Since, all inputs are fixed, we can now handle this using case 1.

Case 3: Suppose f has two variables that are alive. Let $x_1 = b$ where $b \in \{0, 1\}$.

Case 3(a): Suppose $x_2 = x_3 = x$. By setting the output of f to $\bar{b}$ we fix the value of x_2, x_3 to $\bar{b}$. This eliminates all the variables from $\mathcal{O}$. Again, this reduces to case 1.

Case 3(b): Suppose $x_2 = x, x_3 = \overline{x}$. This fixes the value of f to b. Hence, setting the output of f to $\bar{b}$ gives a string outside the range.

Case 4: Suppose all the three variables of f are alive. Note that since $|\mathcal{O}| \leq n-1$ and $m > n$ there must be at least two more functions outside $\mathcal{O}$. If any of these functions satisfy the above cases then we have already found a solution to the problem. Therefore, we consider the case when all three functions have all the three variables that are alive. By PHP, there exist two functions at least two of whose inputs are set to $x(\overline{x})$. This fixes the output of these functions to $x(\overline{x})$. By setting one of outputs to 1 and the other to 0 yields a string outside the range.

Thus, if there exist two function in C which have at least two common inputs then as described above, we already have a solution to the range avoidance problem. Otherwise, we have an instance of ONE-INTERSECT-MAJ$_3$-AVOID. Note that the above steps can be done in polynomial time. Hence, it suffices to show a polynomial time algorithm for this case. ☐

Having described all the ingredients of the proof, we are now ready to prove our first algorithm for MAJ$_3$-AVOID (which works only for quadratic stretch $m > cn^2$). We apply step 1 to obtain a sub-circuit C_K whose outputs form a cluster K. Next we observe that any two functions in C_K share at most two input variables since otherwise we can trivially obtain a string outside the range by assigning opposite values to the corresponding output bits. By step 2, we reduce our problem to an instance of ONE-INTERSECT-MAJ$_3$-AVOID in polynomial time. Since $m > cn^2$, theorem 3.10 gives a polynomial time algorithm for solving ONE-INTERSECT-MAJ$_3$-AVOID.

This gives a polynomial time algorithm for solving MONOTONE-NC$^0_3$-AVOID with quadratic stretch, thus completing the proof of the following theorem from the introduction.

Theorem 1.4. MONOTONE-NC$^0_3$-AVOID *when $m > cn^2$ for any constant c can be solved in deterministic polynomial time.*

Improved Polynomial Time Algorithms for MONOTONE-NC$^0_3$-AVOID **with Linear Stretch:** In the previous section, we saw that using wicket as our candidate for forbidden sub-hypergraph, works for quadratic stretch. In order improve this bound on the stretch requirement, we shall use loose a different forbidden sub-hypergraphs which are $\mathcal{X}_{2\ell}$-cycles. It suffices to show that the structure has an edge-coloring which is not MAJ-coloring and that the extremal number for the structure is not too large.

First, we show that there exists an edge-coloring of $\mathcal{X}_{2\ell}$ which is not MAJ-coloring.

Lemma 4.5. *Let $\mathcal{H}$ be $\mathcal{X}_{2\ell}$-cycle. Then there exists an edge-coloring of $\mathcal{H}$ which is not* MAJ*-coloring.*

Proof. Let $v_1 e_1 v_2 \ldots v_{2\ell} e_{2\ell} v_1$ be the $\mathcal{X}_{2\ell}$ cycle with (e_i, e_j) as χ-structure where $i \equiv 0 \mod 2$, $j \equiv 1 \mod 2$. Consider the following edge-coloring-

$$\Gamma(e_i) = \begin{cases} R, & \text{if } i \equiv 0 \mod 2 \\ B, & \text{if } i \equiv 1 \mod 2 \end{cases}$$

Let $e_i \cap e_j = \{x\}$. First consider the case when $\Pi(x) = R$. This forces the color of the other two endpoints of e_j to B. Iteratively, we get $\Pi(v_j) = B$ iff $j \equiv 1 \mod 2$. This implies $\Gamma(e_i) = B$ which is a contradiction. The other case when $\Pi(x) = B$ is similar. □

The following is an easy corollary of lemma 4.5.

Corollary 4.6. *Let $\mathcal{H}$ be a loose $\mathcal{X}_{2\ell}$-cycle. Then there exists an edge-coloring of $\mathcal{H}$ which is not a* MAJ-*coloring.*

In the following theorem, we show an extremal bound for the loose $C_{2\ell}$ in a connected 3-uniform linear hypergraph.

Theorem 1.6. *Any connected 3-uniform linear hypergraph with $m > n$ edges must contain a loose $\mathcal{X}_{2\ell}$ cycle.*

Proof. Let $\mathcal{H}$ a connected 3-uniform linear hypergraph. It suffices to show that there is a loose $\mathcal{X}_{2\ell}$ cycle in $\mathcal{H}$. Towards this, we construct *block graph* G corresponding to H as follows: let the set of m hyperedges in H be the vertices of G and add an edge (f, g) if the hyperedges f, g intersect at a vertex. Observe that G is connected. Now we shall argue that finding a copy of loose $\mathcal{X}_{2\ell}$ cycle in $\mathcal{H}$ is equivalent to finding a subgraph G' in G where G' consists of an edge (f, f') and distinct odd walks from g to g' and h to h' without using edge (f, f'), where g, h and g', h' are neighbors of f, f' respectively in G. The equivalence follows from the observation that a walk of length ℓ corresponds to a walk of length $\ell - 1$ in G. Thus, our task is to show that there must be a G' in G.

Observe that G has m vertices and $3m$ edges. Hence, G must contain a cycle Q. Let $e = (f, f')$ be an edge in Q. Now we would like to show that $G \setminus e$ contains G'. We will show the argument for the g to h walk, the other argument is symmetric. Note that $G \setminus e$ is still connected. Hence, there must be a path P from g, h. Now $P' = (f, g) \cup P \cup (h, f')$ is a path from f to f'. If P' is an odd length path then we have our desired path from f to f' otherwise we shall modify P' to obtain a walk of odd length from f to f'. We observe that the average degree of a vertex in G is $\frac{3m}{n}$. Hence, there a vertex with minimum degree $\frac{3m}{n}$. Observe that these vertices form a clique in G. Let f'' be one of these functions. Note that f'' has degree at least $3m/n > 3$ in G. Hence, there must be an odd cycle C' containing f''. Now, there must be a path Q' from g to f'' owing to connectivity of G_C. Thus, $qQ'f''C'f''Q'gPg'$ is an odd walk from g to g'. Adding edges $(f, g), (f', g')$ to Q' gives an odd walk from f to f'. The other walk containing h, h' can be obtained similarly. Hence there is a copy of G'(loose $\mathcal{X}_{2\ell}$ cycle) in $G(\mathcal{H})$. This completes the proof. □

Finally, combining all the results above, we shall now prove our first algorithm for MONOTONE-NC^0_3-AVOID with linear stretch.

Theorem 1.7. ***(Main Theorem)*** *For $m > n$,* MONOTONE-NC^0_3-AVOID *can be solved in deterministic polynomial time.*

Proof. By lemma 4.2 and lemma 4.3, we first reduce the circuit to a sub-circuit corresponding to cluster, and then further to the case of ONE-INTERSECT-MAJ$_3$-AVOID. Hence, it suffices to solve ONE-INTERSECT-MAJ$_3$-AVOID. By theorem 1.6 we know that there exists a loose $\mathcal{X}_{2\ell}$ in the hypergraph corresponding to the given circuit. It suffices to show that we can find loose $\mathcal{X}_{2\ell}$ in $\mathcal{H}$ in polynomial time. By corollary 4.6 and theorem 3.6 we get a polynomial time algorithm to solve MAJ$_3$-AVOID. Indeed to find G' in the proof of theorem 1.6, we need to find the paths P, Q', and a high degree vertex f'', which can be done in polynomial time. Hence, we can find the G' in polynomial time. ☐

5 Polynomial time Algorithm for SYMMETRIC-NC$^0_3$-AVOID

A natural restriction of the range avoidance problem would be when all the functions are symmetric. We call this the SYMMETRIC-AVOID problem. In this section, we use our results for MONOTONE-NC$^0_3$-AVOID to solve SYMMETRIC-NC$^0_3$-AVOID. We look at all possible symmetric functions over three bits and handle each case. We will need the following lemma, which shows that range avoidance problem is easy when all the functions compute parity.

Lemma 5.1. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a circuit such that each output function is a parity function and $m > n$. Then there is a deterministic polynomial time algorithm to find a string outside the range of C.*

Proof. We note that since each output function computes parity of input bits. These are linear functions. We can express this in the form of matrix M of dimensions $m \times n$ as follows:

$$M_{ij} = \begin{cases} 1 & \text{if } f_i \text{ depends on variable } x_j \\ 0 & \text{otherwise} \end{cases}$$

Let $x = (x_1, \dots x_n)$. Observe that the Mx is the range of circuit C. Let $C_1, \dots C_n$ be the column vectors of M. We have that the column space of C_i is the range of C. Let $e_1, \dots e_m$ be the standard basis vectors. We claim that there exists an e_i which is outside the range of C. Otherwise the column-space covers each m-bit string which is a contradiction since a string outside the range is guaranteed by $m > n$. We observe that $e_i \in Range(C) \iff \{C_1, \dots C_n, e_i\}$ is a linearly dependent. Since, we can check whether a set of vectors is linearly independent in polynomial time, we can find an e_j which is outside the range of C. ☐

We can extend the above idea to work with MOD$_3$ functions with different residues. We define the MOD$_3^i$ function for $i \in \{0,1,2\}$ as follows:

$$\text{MOD}_3^i(x) = \begin{cases} 1 & \text{if } x_1 + x_2 + x_3 \equiv i \mod 3 \\ 0 & \text{otherwise} \end{cases}$$

Lemma 5.2. *Let $C : \{0,1\}^n \to \{0,1\}^m$ be a circuit such that each output function is a* MOD$_3^i$ *function for $i \in \{0,1,2\}$ and $m > n$. Then there is a deterministic polynomial time algorithm to find a string outside the range of C.*

Algorithm 1 Algorithm for solving MAJ$_3$-AVOID on input $C : \{0,1\}^n \to \{0,1\}^m$

Step 1: Obtain a cluster K such that $|I(K)| < |K|$ ▷ by observation 4.2
Step 2: Reduction from TWO-INTERSECT-MAJ$_3$-AVOID to ONE-INTERSECT-MAJ$_3$-AVOID
if $f_1, f_2 \in K$ such that $I(f_1) = \{x_1, x_2, x_3\}$ and $I(f_2) = \{x_2, x_3, x_4\}$ **then**
  Set $f_1 = 1, f_2 = 0$ and $x_1 = 1, x_2 = 0, x_4 = \neg x_3$, $\mathcal{O} = \{f_1, f_2\}$
  repeat
    Consider $g \in K \setminus \mathcal{O}$ such that $I(g) \neq \emptyset$
    if $|I(g) \cap I(\mathcal{O})| = 2$ **then**
      Let $I(g) = \{z_1, z_2, z_3\}$ and $z_3 = I(g) \setminus I(\mathcal{O})$. Let $b \in \{0,1\}$
      if Both z_1, z_2 are not alive: **then**
        If $z_1 = z_2 = b$ By setting $g = \overline{b}$, we obtain $y \notin \mathsf{Range}(C)$
        If $z_1 = b$, $z_2 = \overline{b}$ then set $g = b$ and $y = b$
      else if Exactly one of z_1, z_2 is alive: say $z_1 = b$, $z_2 = x$ **then**
        Setting $g = z_3 = \overline{b}$ eliminates all the variables
      else if Both z_1, z_2 are alive: that is $z_1 = x, z_2 = \neg x$ **then**
        We set $g = z_3 = 0$ so that the number of variables in the cluster is at most 1.
      end if
    else if $I(g) \cap I(\mathcal{O}) = \{z_1\}$ **then**
      If $z_1 = b \in \{0,1\}$, then Set g to $\overline{b}$ and $z_2 = z_3 = \overline{b}$
      If $z_1 = x$(alive) then set g to $\overline{x}$ and $z_2 = z_3 = \overline{x}$
      Update the cluster $\mathcal{O} = \mathcal{O} \cup \{g\}$ and $I(\mathcal{O}) = I(\mathcal{O}) \cup I(g)$
    end if
  until $\exists g \notin \mathcal{O}$ such that $I(g) \cap I(\mathcal{O}) \neq \emptyset$
end if
$\exists h_1, h_2, h_3 \notin \mathcal{O}$ such that $I(f) = \{w_1, w_2, w_3\} \subseteq I(\mathcal{O})$ since $|\mathcal{O}| \leq n-2$ and $|I(\mathcal{O})| = n$.
if $\nexists$ a variable which is alive **then**
  Setting the output of h_1 to $\overline{b}$ gives a string outside the range.
else if $\exists$ exactly one variable w_1 which is alive **then**
  if $w_2 = w_3 = b$ **then**
    Setting the output of h_1 to $\overline{b}$ gives a string outside the range.
  else if $w_2 = b, w_3 = \overline{b}$ **then**
    Set the output of h_1 to b, which fixes w_1 to b.
    We have $h_2 \notin \mathcal{O}$ and $I(h_2) \subseteq I(\mathcal{O})$. Same as previous case.
  end if
else if h_1 has two variables that are alive and $w_3 = b$ **then**
  if $w_1 = w_2 = x$ **then**
    We set the output of h_1 to $\overline{b}$. This fixes all input variable(case 1).
  else if $w_1 = x, w_2 = \overline{x}$ **then**
    Setting the output of h_1 to $\overline{b}$ gives a string outside the range.
  end if
else if All three variables are alive **then**
  $\exists\, h_i, h_j$ whose inputs are set the same. Setting h_i to 1 and h_j to 0 gives $y \notin \mathsf{Range}(C)$.
end if
Step 3: Polynomial time algorithm for ONE-INTERSECT-MAJ$_3$-AVOID
Find a loose $\mathcal{X}_{2\ell}$ in $\mathcal{H}_C$ ▷ by theorem 1.6
Output $y \notin \mathsf{Range}(C)$ ▷ by corollary 3.6

Proof. We express these linear functions in terms of matrix M with dimensions $m \times (n+1)$ as follows:

$$M_{jk} = \begin{cases} 1 & \text{if } f_j \text{ depends on variable } x_k \text{ and } k \in [n] \\ i & \text{if } f_j \text{ is an MOD}_3^i \text{ function and } k = n+1 \\ 0 & \text{otherwise} \end{cases}$$

Let $C_1, \dots C_{n+1}$ be the column vectors of M. Then the column span of $\{C_1, \dots C_n\} \oplus_3 C_{n+1}$ is the range of C where $\oplus_3$ denotes bitwise MOD_3. As earlier let $e_1, \dots e_m$ denote the standard basis vectors. One of the e_i must be outside the range of C since otherwise every m-bit string is in the range of C, which is a contradiction. We observe that $e_i \in \mathit{Range}(C) \iff \{C_1, C_2 \dots C_n, e_i \oplus_3 C_{n+1} \oplus_3 C_{n+1}\}$ is linearly dependent. For each of the vectors $e_1, e_2, \dots e_m$ we can check the linear dependence in polynomial time. Hence, we can obtain a string outside the range of C in polynomial time. □

Equipped with lemma 5.1 and lemma 5.2, we show a polynomial time algorithm for SYMMETRIC-NC^0_3-AVOID using a reduction to the MAJ_3-AVOID problem.

Theorem 5.3. *There is a polynomial time algorithm to solve* SYMMETRIC-NC^0_3-AVOID *when* $m > 8n$.

Proof. It suffices to show that SYMMETRIC-NC^0_3-AVOID reduces to MAJ_3-AVOID when $m > 8n$ since we have a polynomial time algorithm to solve MAJ_3-AVOID when $m > n$.

We observe that there are only six types of symmetric functions possible on three bits: the MAJ_3 functions, the parity functions(MOD_2), the modulo 3 functions with different residues MOD_3^i and their respective negations

We assume that there are at least $4n$ functions of the form $\text{MAJ}_3, \text{MOD}_2, \text{MOD}_3^i$. The other case when there are more than $4n$ negations of these functions can be handled by flipping the output bit values. Again, by PHP the following cases arise:

Case 1: Suppose there are more than $2n$ functions of the form MOD_2 or MOD_3^i. By repeated application of PHP we get the further cases:

- **Case 1(a):** There are more than n parity functions in the circuit. By lemma 5.1, we can find a string outside the range in polynomial time.
- **Case 1(b):** Consider the other case when the number of MOD_3^i functions is more than n. In this case we can obtain a string which is outside the range of the circuit in polynomial time using lemma 5.2.

Case 2: There are more than $2n$ MAJ_3 functions. It suffices to solve the range avoidance problem for the MAJ_3-AVOID problem.

□

Thus, we have a polynomial time algorithm to solve SYMMETRIC-NC^0_3-AVOID in polynomial time when $m > 8n$.

6 Conclusion

We described the formulation of special cases AVOID in terms of Turan-type problems in k-uniform hypergraphs, and demonstrated some applications to solve monotone versions of AVOID under depth restrictions for the circuit. We exhibited several different fixed hypergraphs $\mathcal{H}'$ - k-cage, weak Fano plane, 3×3 grid, $(3,3)$-butterfly, $(3,3)$-odd kite which can be also used to derive polynomial time algorithms for ONE-INTERSECT-MAJ$_3$-AVOID which in turn is used to solve MONOTONE-NC_3^0-AVOID when the strech is quadratic. Finally, the improvement to linear stretch (theorem 1.7) comes from using loose $\mathcal{X}_{2\ell}$ cycles as our hypergraph. We note that, this is not a fixed sized hypergraph, however, we can find it in polynomial time. We also obtain a polynomial time algorithm for SYMMETRIC-NC_3^0-AVOID with linear stretch. Prior to this work, the best known polynomial time algorithm MONOTONE-NC_3^0-AVOID and SYMMETRIC-NC_3^0-AVOID was via the algorithm for NC_3^0-AVOID which requires $m > \frac{n^2}{\log n}$ [GGNS23]. We extend our framework to solve ONE-INTERSECT-MAJ$_k$-AVOID for linear and quadratic stretch respectively. [GGNS23] give a polynomial time algorithm for solving NC_k^0-AVOID when $m > \frac{n^{k-1}}{\log n}$. it would be interesting to improve the stretch using the hypergraph framework even for monotone NC_6^0 circuits, which would in turn give an improvement for the case of NC_3^0-AVOID.

References

[BES73] WG Brown, Pál Erdös, and Vera Sós. Some Extremal Problems on Gamma-graphs. Academic Press, Inc, 1973.

[CCGJ18] Claton Collier-Cartaino, Nathan Graber, and Tao Jiang. Linear Turán Numbers of Linear Cycles and Cycle-Complete Ramsey Numbers. *Combinatorics, Probability and Computing*, 27(3):358–386, 2018.

[CHR] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric Exponential Time Requires Near-Maximum Circuit Size, year = 2024. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, page 1990–1999.

[FR13] Zoltán Füredi and Miklós Ruszinkó. Uniform hypergraphs containing no grids. *Advances in Mathematics*, 240:302–324, 2013.

[FS24] Jakob Führer and Jozsef Solymosi. Caps and Wickets. Technical report, 2024.

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range Avoidance for Constant Depth Circuits: Hardness and Algorithms. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 65:1–65:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[GL12] Ervin Gyori and Nathan Lemons. Hypergraphs with no cycle of a given length. *Combinatorics, Probability and Computing*, 21(1–2):193–201, 2012.

[GL21] W. T. Gowers and J. Long. The length of an s-increasing sequence of r-tuples. *Combinatorics, Probability and Computing*, 30(5):686–721, 2021.

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:21, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GS22] András Gyárfás and Gábor N. Sárközy. The linear Turán number of small triple systems or why is the wicket interesting? *Discrete Mathematics*, 345(11):113025, 2022.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability Obfuscation, Range Avoidance, and Bounded Arithmetic. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, page 1076–1089, 2023.

[Kee11] Peter Keevash. Hypergraph Turán Problems. *Surveys in Combinatorics*, 06 2011.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total Functions in the Polynomial Hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[Kor22] Oliver Korten. The Hardest Explicit Construction. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 433–444, 2022.

[KPI25] Oliver Korten, Toniann Pitassi, and Russell Impagliazzo. Stronger Cell Probe Lower Bounds via Local PRGs. Technical Report 30, Electronic Colloquium on Computational Complexity (ECCC), March 2025.

[KS04] Peter Keevash and Benny Sudakov. The Turán Number Of The Fano Plane. *Combinatorica*, 25, 03 2004.

[Li24] Zeyong Li. Symmetric Exponential Time Requires Near-Maximum Circuit Size: Simplified, Truly Uniform. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, 2024.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the Range Avoidance Problem for Circuits. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 640–650. IEEE, 2022.

[Sol24] Jozsef Solymosi. Wickets in 3-uniform hypergraphs. *Discrete Mathematics*, 347(6):114029, 2024.

[ZBW24] Lin-Peng Zhang, Hajo Broersma, and Ligong Wang. A Note on Generalized Crowns in Linear r-graphs. Technical report, 2024. arxiv : 2401.12339.

A Appendix

A.1 Special cases of MONOTONE-AVOID

We now describe some special cases of the monotone range avoidance problem that are simpler than the general case of AVOID. We consider the depth 1 sub-circuit obtained by viewing circuits as a composition of two circuits. To denote this, for the rest of this discussion, we assume D is a monotone circuit of depth d. For $j \leq d$, define D_j to be the sub-circuit of C obtained by removing all the gates and variables below depth j and $\widehat{D_j}$ be the sub-circuit of C obtained by deleting all the gates above depth j. Let m_j denote the number of gates at depth j for circuit C. Hence $D_j : \{0,1\}^{m_j} \to \{0,1\}^m$ and $\widehat{D_j} : \{0,1\}^n \to \{0,1\}^{m_j}$. By definition, $D = D_j \circ \widehat{D_j}$. We show below that when $j \leq 2$, the problem is easier under some constraints for m_1 and m_2 respectively.

Proposition A.1. *If $m > m_1$ then there is a deterministic polynomial time algorithm that finds a string outside the range of the circuit.*

Proof. Since $m > m_1$, we have that D_1 is a valid instance of the AVOID problem. Observe that, D_1 is a depth 1 monotone circuit. We claim that $y \notin \mathsf{Range}(D^1) \implies y \notin \mathsf{Range}(D)$. This follows from the fact that if no input setting $x \in \{0,1\}^{m_1}$ in circuit D^1 can produce a $y \in \{0,1\}^m$, then no input assignment of n variables in circuit D can ever produce y.

We now show how to find a y outside the range of D_1 in deterministic polynomial time. Hence, when $m > m_1$ there is a polynomial time algorithm for AVOID. Similar to the approach in [GLW22], the idea is that we inductively solve the range avoidance problem for smaller circuits. Consider the first output bit y_1 (which is a function f_1 of the input variables). We consider the following cases based on the type of functions:

Case 1: Suppose f is a constant function. Without loss of generality assume f is the constant zero function. Setting $y_1 = 1$ and the remaining output bits to arbitrary Boolean values, we obtain a string y which is outside the range of D_1.

Case 2: Suppose f is the $\wedge$ of some variables. In this case, we set output bit $y_1 = 1$ and variables feeding into this gate to be 1. Thus, we obtain a smaller circuit $D_1' : \{0,1\}^{m_j - 1} \to \{0,1\}^{m-1}$.

Case 3: Suppose f is the $\vee$ of some variables. This case can be handled similar to case 2. Setting output bit $y_1 = 0$ and variables feeding into this gate to be 0, we obtain a smaller circuit $D_1' : \{0,1\}^{m_j - 1} \to \{0,1\}^{m-1}$.

After n steps, we obtain a circuit $D_1'' : \{0,1\}^0 \to \{0,1\}^{m - m_j}$. The value of function f_{m_j+1} is fixed by the input assignments. Flipping this value, we obtain a string outside the range of D_1. We note that this can be done in $O(m)$ time. □

Even if $m < m_1$, m is much larger compared to m_2, there is $\mathsf{FP}^{\mathsf{NP}}$ algorithm for MONOTONE-AVOID where s is the size of the circuit.

Proposition A.2. *If $m > m_2^{\omega(\sqrt{m_2 \log s})}$ there is an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for MONOTONE-AVOID where s is the size of the circuit.*

Proof. We show that if $m_2 < \frac{m}{2}$, then the problem can be reduced in polynomial time to CNF-AVOID. If in addition, $m > m_2^{\omega(\sqrt{m_2 \log s})}$, we can use $\mathsf{FP}^{\mathsf{NP}}$ algorithm due to [GLW22] to solve the CNF-AVOID instance and this implies the above proposition.

The reduction is as follows. Let F_1, F_2 be the set of all $\wedge$ gates and all $\vee$ gates respectively in the output layer. By pigeon hole principle, we have that either $|F_1| \geq \frac{m}{2}$ or $|F_2| \geq \frac{m}{2}$. Without loss of generality assume $|F_1| \geq m/2$. Since $m_2 < \frac{m}{2}$, it suffices to solve the problem for D' obtained from D_2 by eliminating the top $\vee$ gates. Since top layer of D' contains only $\wedge$ gates, we can assume that there are only $\vee$ gates in the second layer of this circuit. Otherwise, we can feed the inputs of this $\wedge$ gate directly to the $\wedge$ gate above it. Now D_2 is a CNF formula. $\square$

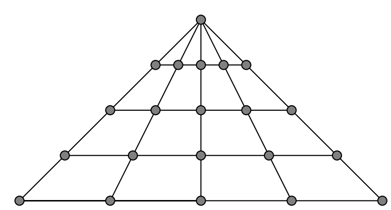
A.2 Alternative Hypergraphs for Solving ONE-INTERSECT-MAJ$_3$-AVOID

app:subsec:structures

In this subsection, we show other choices of hypergraphs that can be used in place of wickets in the algorithm above for solving ONE-INTERSECT-MAJ$_3$-AVOID.

Alternative 1 : k-cage: We define the graph class as below:

Definition A.3. A k-cage is a k-uniform linear hypergraph $\mathcal{H}(V, E)$ with edge set $E = E_1 \cup E_2$ such that E_1 consists of pairwsise disjoint edges which intersect exactly at $w \in V$ and E_2 consists of $k-1$ edges that are pairwise disjoint over $V \setminus \{w\}$. The picture shown is the 5-cage.



We will show that there exists an edge-coloring Γ that is not a MAJ-coloring.

Lemma A.4. *Let $\mathcal{H}$ be a k-cage. There exists an edge coloring $\Gamma : E \to \{R, B\}$ which is not a* MAJ-*coloring. Furthermore, we can find such a coloring in polynomial time.*

Proof. Let $\mathcal{H}$ be a k-cage with k edge-set $E = E_1 \cup E_2$, where E_1 is the set of k-edges intersecting in vertex w and E_2 is the set of $k-1$ pairwise disjoint edges over $V \setminus \{w\}$. We will show t the following Γ is not a MAJ-coloring:

$$\Gamma(e) = \begin{cases} B & \text{if } e \in E_1 \\ R & \text{otherwise} \end{cases}$$

We consider the following cases based on the parity of k:

Case 1: Suppose k is odd. Observe that for an edge in E_2 to be colored R at least $\lfloor \frac{k}{2} \rfloor + 1$ of its vertices should be colored R. Since the edges in E_2 are pairwise disjoint, we have that there should be at least $k\left(\lfloor \frac{k}{2} \rfloor + 1\right)$ vertices in $V \setminus \{w\}$ that are colored R. Similarly, for any edge in $e \in E_1$ to be colored B, we have that at least $\lfloor \frac{k}{2} \rfloor$ vertices in $e \setminus \{w\}$ should be colored B. Since, any two edges in E_1 intersect exactly at vertex w, we have that at least $\left(\lfloor \frac{k}{2} \rfloor\right)(k-1)$ vertices in $V \setminus \{w\}$ that should be colored R. Thus, totally there must be at least $\left(\left\lfloor\frac{k}{2}\right\rfloor\right)(k-1)+\left(\left\lfloor\frac{k}{2}\right\rfloor+1\right)(k)=k^2+k-\lfloor k/2\rfloor>k(k-1)$ vertices in $V\setminus\{w\}$. Since, $|V\setminus\{w\}|=k(k-1)$ we get a contradiction.

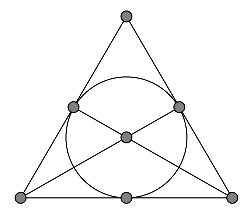
Case 2**:** Suppose k is even. As earlier for the edges in E_2 to be colored R there must be at least $k\left(\frac{k}{2}+1\right)$ vertices in $V\setminus\{w\}$ that are colored R. Since k is odd, for any edge in $e\in E_1$ to be colored B, we have that at least $\frac{k}{2}-1$ vertices in $e\setminus\{w\}$ should be colored B. Hence, at least $\left(\frac{k}{2}-1\right)(k-1)$ vertices in $V\setminus\{w\}$ should be colored R. Thus, totally there must be at least $\left(\frac{k}{2}+1\right)(k)+\left(\frac{k}{2}-1\right)(k-1)=k^2-k/2-1$ vertices in $V\setminus\{w\}>k(k-1)$. Since, $|V\setminus\{w\}|=k(k-1)$ we get a contradiction.

In either case, we show that Γ is not a MAJ-coloring of $\mathcal{H}$. □

Alternative 2 : Weak Fano plane: This structure is similar to the fano plane that was defined in section 2 with one hyperedge removed.

Definition A.5. A weak Fano plane is a 3-uniform linear hypergraph which is isomorphic to the hypergraph $\mathcal{H}(V,E)$ with vertex set $V=[7]$ and edge set

$$E=\{\{1,2,3\},\{3,4,5\},\{1,5,6\},\{3,6,7\},\{2,5,7\},\{2,4,6\}\}$$



We will show that there $\Gamma:E\to\{R,B\}$ which is not a MAJ-coloring.

Lemma A.6. *Let $\mathcal{H}$ be a weak Fano plane hypergraph. Then there is an edge-coloring $\Gamma:E\to\{R,B\}$ which is not a* MAJ*-coloring. Furthermore, we can find this coloring in polynomial time.*

Proof. Let $E_1=\{e|1\in e\}$. We will show that the following Γ is not a MAJ-coloring.

$$\Gamma(e)=\begin{cases}R & \text{if } e\in E_1\\ B & \text{otherwise}\end{cases}$$

We consider the following cases:

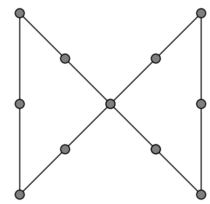
Case 1**:** Suppose $\Pi(1)=B$. Then for any edge $e\in E_1$ to be colored R, it must be that the other two vertices in e should be R. In particular, we have $\Pi(3)=\Pi(5)=R$. This implies $\Gamma(\{3,4,5\})=R$ which is a contradiction.

Case 2**:** Suppose $\Pi(1)=R$. Then for an edge $e\in E_1$ to be colored R there must be a vertex pair $\{i,j\}\in\{\{2,5\},\{2,6)\},\{3,5\},\{3,6\}\}$ such that $\Pi(i)=\Pi(j)=R$. We will obtain a contradiction in each case: Suppose $\{i,j\}=\{2,5\}$ then $\Gamma(\{2,5,7\})=R$ which is a contradiction. For $\{i,j\}$ being one of $\{2,6\},\{3,5\},\{3,6\}$ we similarly obtain contradictions for the coloring of the edges $\{2,4,6\},\{3,4,6\},\{3,7,6\}$ respectively.

Hence, Γ is not a MAJ-coloring of $\mathcal{H}$. □

Alternative 3 : (k,ℓ)**-odd butterfly:** A 3-uniform loose cycle of length k, $C := u_1e_1u_2\dots u_ke_ku_1$ is a hypergraph over the vertex set $\{u_1,\dots,u_k\}\cup\{v_1,\dots v_k\}$ such that $u_i,u_{i+1}\in e_i$ and $e_i=(u_i,v_i,u_{i+1})$ where $u_{k+1}=u_1$.

Definition A.7. A (k,ℓ)-odd butterfly is a 3-uniform hypergraph over $E=\{e_1\dots e_k\}\cup\{f_1,\dots f_\ell\}$ consisting of two loose cycles $C_1 := u_1e_1u_2e_2\dots u_ke_ku_1$ and $C_2 := u_1f_1v_2f_2\dots v_\ell f_\ell u_1$ where k,ℓ are odd. The figure demonstrates a $(3,3)$-butterfly.



In the following lemma we show an edge coloring function Γ which is not a MAJ-coloring.

Lemma A.8. *Let $\mathcal{H}$ be a (k,ℓ)-odd butterfly. Then there is an edge coloring $\Gamma : E(\mathcal{H})\to\{R,B\}$ which is not* MAJ*-coloring. Furthermore, we can find this coloring in polynomial time.*

Proof. Let C_1,C_2 be the two loose cycles of order k in $\mathcal{H}$. Let $C_1 := u_1e_1u_2e_2\dots u_ke_ku_1$ and $C_2 := u_1f_1v_2f_2\dots v_\ell f_\ell u_1$. We will show that the following edge-coloring $\Gamma : E\to\{R,B\}$ is not a MAJ-coloring.

$$\Gamma(e_i)=\begin{cases}R, & \text{if } i \text{ is even}\\ B, & \text{otherwise}\end{cases}$$

$$\Gamma(f_i)=\begin{cases}R, & \text{if } i \text{ is odd}\\ B, & \text{otherwise}\end{cases}$$

where $i\in[k]$. First, we will show the following claim:

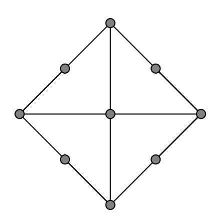
Claim A.9. *Let $\Gamma : E\to\{R,B\}$ be a* MAJ*-coloring under vertex coloring $\Pi : V\to\{R,B\}$ such that $\Pi(u_i)=c$ and $\Gamma(e_i)=\bar{c}$ where $c\in\{R,B\}$ and $\bar{c}=\{R,B\}\setminus\{c\}$. Then $\Pi(u_{i+1})=\bar{c}$.*

Proof. Wlog let $\Pi(u_i)=R$. Since Γ is a MAJ-coloring satisfying $\Gamma(e_i)=B$, the other two vertices in e_i must be B. In particular we have that $\Pi(u_{i+1})=B$. □

Suppose $\Pi(u_1)=R$. Then by claim A.9 we have that $u_k=R$. We have $\Gamma(e_k)=B$. This implies $\Pi(u_1)=B$ which a contradiction. Similarly, if $\Pi(u_1)=B$ then we can repeatedly apply the claim A.9 for C_2 to obtain a contradiction. □

Alternative 4 : (k,ℓ)**-odd kite:** We define the hypergraph as follows:

Definition A.10. A (k,ℓ)-odd kite is a 3-uniform linear hypergraph with edge set $E=\{e_1,\dots e_k\}\cup\{f_1,\dots f_k\}\cup\{e'\}$ consisting of two loose odd cycles $C_1=u_1e_1u_2\dots u_ke_ku_1$, $C_2=u_1f_1v_2f_2\dots f_{\ell-1}u_kf_\ell u_1$ where $e_k=f_\ell=\{u_0,u_1,u_k\}$, $e'=\{u_0,u_i,v_j\}$ such that $i\in\{2,\dots k-1\}$, $j\in\{2,3\dots\ell-1\}$ and $i\equiv j \mod 2$. The figure shows a $(3,3)$-odd kite.



We will show an edge coloring of (k,ℓ) which is not a valid MAJ-coloring.

Lemma A.11. *Let $\mathcal{H}$ be a (k,ℓ)-odd kite. Then there exists an edge-coloring $\Gamma : E \to \{R, B\}$ which is not a valid* MAJ*-coloring. Furthermore, we can find such a coloring in polynomial time.*

Proof. We will show that the following edge-coloring Γ is not a valid edge-coloring:

$$\Gamma(e) = \begin{cases} B, & \text{if } e = e_i \in \{e_1, \dots e_k\} \text{ and } i \text{ is odd} \\ R, & \text{if } e = e_i \in \{e_1, \dots e_k\} \text{ and } i \text{ is even} \\ R, & \text{if } e = f_j \in \{f_1, \dots f_{\ell-1}\} \text{ and } j \text{ is odd} \\ B, & \text{if } e = f_j \in \{f_1, \dots f_{\ell-1}\} \text{ and } j \text{ is even} \\ R, & \text{if } e = e' = \{u_0, u_i, v_j\} \text{ and } i \text{ is even} \\ B, & \text{if } e = e' = \{u_0, u_i, v_j\} \text{ and } i \text{ is odd} \end{cases}$$

We consider the following cases based on the colors of vertices u_1, u_k:

Case 1**:** Suppose $\Pi(u_1) = \Pi(u_k) = R$. Since two of its vertices are colored R, this implies that color of edge e_k is R, which is a contradiction.

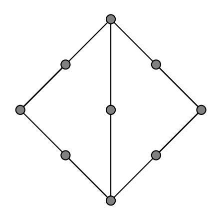
Case 2**:** Wlog let $\Pi(u_1) = R, \Pi(u_k) = B$. By repeated application of claim A.9 in cycle C_1 we get that $\Pi(u_k) = R$, which is again a contradiction. The other can be handles similarly by applying claim A.9 on cycle C_2.

Case 3**:** Suppose $\Pi(u_1) = \Pi(u_k) = B$. By repeatedly applying claim A.9 in cycles C_1, C_2 we get that $\Pi(u_i) = B \iff i \equiv 0 \mod 2$ and $\Pi(v_j) = B \iff j \equiv 0 \mod 2$. This implies that $\Gamma(e') = B \iff i \equiv 0 \mod 2$, which is a contradiction.

Since, in either case we obtain a contradiction, Γ is not a valid MAJ-coloring. □

A.3 Counter Example graphs among the $(9,5)$-hypergraphs

In continuation with the discussion towards the introduction, we write down the explicit example of a $(9,5)$-graph which does not lead to a contradiction as used in our arguments. It is easy to verify that every edge coloring is a MAJ-coloring for this hypergraph. This example explains why the power-bound conjecture in [GL21] is not sufficient to provide us the required improvements for the relation between m and n in our proposed algorithm for solving MAJ_3-AVOID and hence MONOTONE-NC^0_3-AVOID.



ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il