6-12-2020

Towards Ryser's Conjecture: Bounds on the Cardinality of Partitioned Intersecting Hypergraphs

Anna E. Dodson
Dartmouth College

Towards Ryser's Conjecture

Bounds on the Cardinality of Partitioned Intersecting Hypergraphs

Anna Dodson

Undergraduate thesis presented for the degree of
Bachelor of Arts in Computer Science

Dartmouth College
Department of Computer Science
Technical Report TR2020-902

June 12, 2020

Contents

Towards Ryser's Conjecture: Bounds on the Cardinality of Partitioned Intersecting Hypergraphs

Anna Dodson

June 12, 2020

Abstract

This work is motivated by the open conjecture concerning the size of a minimum vertex cover in a partitioned hypergraph. In an r-uniform r-partite hypergraph, the size of the minimum vertex cover C is conjectured to be related to the size of its maximum matching M by the relation ($|C| \leq (r-1)|M|$). In fact it is not known whether this conjecture holds when $|M| = 1$. We consider r-partite hypergraphs with maximal matching size $|M| = 1$, and pose a novel algorithmic approach to finding a vertex cover of size $(r-1)$ in this case. We define a *reactive hypergraph* to be a back-and-forth algorithm for a hypergraph which chooses new edges in response to a choice of vertex cover, and prove that this algorithm terminates for all hypergraphs of orders $r = 3$ and 4. We introduce the idea of optimizing the size of the reactive hypergraph and find that the reactive hypergraph terminates for $r = 5...20$. We then consider the case where the intersection of any two edges is exactly 1. We prove bounds on the size of this 1-intersecting hypergraph and relate the 1-intersecting hypergraph maximization problem to mutually orthogonal Latin squares. We propose a generative algorithm for 1-intersecting hypergraphs of maximal size for prime powers $r - 1 = p^d$ under the constraint $pd + 1$ is also a prime power of the same form, and therefore pose a new generating algorithm for MOLS based upon intersecting hypergraphs. We prove this algorithm generates a valid set of mutually orthogonal Latin squares and prove the construction guarantees certain symmetric properties. We conclude that a conjecture by Lovász [1], that the inequality in Ryser's Conjecture cannot be improved when $(r-1)$ is a prime power, is correct for the 1-intersecting hypergraph of prime power orders.

1 Introduction

A r-uniform hypergraph $G = (V, E)$ consists of a collection of vertices $v \in V$ and a collection of edges $e \in E$ where an edge corresponds to a r element subset of V. A *vertex cover* of G is a subset of vertices where each edge contains at least one vertex in the cover. A *minimum vertex cover* is a cover with the minimum number of possible vertices. Finding the minimum vertex cover in a hypergraph is a known NP-complete problem. A *matching* is a selection of hyperedges such that no two have any vertex in common, and a *maximum matching* is a matching such that the cardinality is maximized.

Let the maximum matching be denoted by M and the minimum vertex cover be denoted by C. König's Theorem [2] states that for a bipartite graph, the maximum matching has the same cardinality as its minimum vertex cover; that is, $|C| = |M|$. An *r-partite hypergraph* is one where the vertices may be divided into groups such that all edges in the hypergraph contain at most one vertex from each group. Ryser's Conjecture follows from here: that for any hypergraph of order r which can be partitioned into r parts such that all edges contain at most one vertex in each part, the cardinality of the minimum vertex cover C is less than or equal to $(r-1)$ times the cardinality of its maximum matching M. To summarize, the bound on the size of the minimum vertex cover in the r-partite hypergraph is posed by Ryser:

Conjecture 2. Ryser's conjecture. For any r-partite hypergraph,

$$|C| \leq (r-1)|M| \quad [3].$$

For any order of hypergraph, it is known that finding a maximal matching or a minimum vertex cover is NP-hard. There has been no polynomial time algorithm for finding the maximal matching or the minimal vertex cover in any hypergraph of order larger than three. Approximation of minimum vertex cover has been shown to be NP-hard, with NP-hardness of obtaining an approximation factor of $(\frac{r}{4} - \epsilon)$ for even r and $(\frac{r}{4} - \frac{1}{4r} - \epsilon)$ for odd r [4].

In the bipartite graph, Hall's Theorem proves the conjecture, and using the Ford-Fulkerson algorithm for maximum flow, we may find a maximum matching which we use to build a maximum alternating forest starting from non-matched vertices in M and from there define the cover [5]. In the hypergraph case, no polynomial time algorithm for vertex cover is known. Ron Aharoni [6] proves the conjecture for tripartite graphs using a topological proof. Thus he is able to show $|C| \leq 2|M|$, but does not pose an algorithm for the vertex cover.

Our posed algorithm follows from a modification of an approximation algorithm for general hypergraphs. We outline the proposed vertex cover algorithm for $|M| = 1$:

- Initialize the solution $C = \{\}$.
- Put all edges in E in a set $E_{remaining}$.
- Add the vertex with the maximum degree in $E_{remaining}$ to the set C. Let the edges incident be $E_{incident}$. $E_{remaining} \leftarrow E_{remaining} - E_{incident}$.
- Repeat until $(r-1)$ vertices have been chosen.

The nonpartitioned hypergraph approximation algorithm adds both vertices in each edge e and continues until $E_{remaining}$ is empty, and has been shown to always find a vertex cover whose size is more than twice the size of minimum possible vertex cover. Our partitioned hypergraph algorithm terminates after $(r-1)$ vertices are selected, and throws an error if the $(r-1)$ vertices do not cover the hypergraph in its entirety. While we do not guarantee that this algorithm finds the minimum vertex cover, if it succeeds, we conclude we can always select a cover which contains $(r-1)$ vertices, showing tightness in Ryser: $|C| = (r-1)$.

To verify the validity of a vertex cover algorithm, we introduce the notion of a *reactive hypergraph*. This kind of hypergraph can be thought of as a back-and-forth game between two players, Alice and Bob. At each time step, Alice poses a candidate vertex cover. Bob then must react to add an edge to the hypergraph which intersects all other edges in the hypergraph, satisfying $|M| = 1$, but escapes this candidate vertex cover. The two players continue like this until no further play is possible. In the case that Alice loses, i.e., the posed vertex cover is not a valid vertex cover and does not cover all edges, we invalidate her candidate vertex cover algorithm. If Bob cannot add any more edges, then her vertex cover algorithm is still a candidate. This is an empirical approach to verifying a polynomial-time vertex cover algorithm.

For small r, we explore all of Bob's possible plays and thus generate all possible reactive hypergraphs. Bob's possible plays include all valid edges which may or may not include newly created vertices, but we limit the enumerations of new vertices at any given time step. We prove in this document that Alice's vertex cover algorithm terminates for $r = 3$. We prove using brute-force methods that Alice's vertex cover algorithm terminates for any hypergraph of order $r = 4$. When $r \geq 5$, the number of possible reactive hypergraphs exceeds a hundred thousand, and it is not computationally feasible to brute-force all possible games. Therefore we pose another kind of hypergraph, the *optimal reactive hypergraph*. In an optimal reactive hypergraph,

edges are added in such a way to attempt to maximize the size of the resulting hypergraph. Bob would like to evade Alice for as long as possible, thereby adding as many edges as possible. Using a 'greedy' approach based on Alice's algorithm for the vertex cover, we find that the algorithm terminates in the optimal reactive hypergraph for $r = 5...20$. It is difficult to quantify how to play optimally; we find that the key to optimal play in the $r = 4$ case is introducing an edge which plays the vertex of second-highest degree, rather than maximum degree, at the midway point in the hypergraph. This may inform further algorithmic work in deepening the size of the optimal reactive hypergraph. In summary of the empirical findings, we create a novel framework for verifying a vertex cover algorithm and find empirically that a novel vertex cover algorithm may find an $(r-1)$-sized vertex cover in $O(r(V+E))$ time.

The question of maximizing the size of an intersecting hypergraph is further explored in the 1-intersecting hypergraph. We define a *k-intersecting hypergraph* as a hypergraph where all edges intersect all other edges k times. In the 1-intersecting hypergraph, we limit $k = 1$, and prove bounds on the hypergraph under this constraint. All edges intersect all other edges exactly once in this class of hypergraph. We show:

Theorem 4. *In a k-intersecting hypergraph with $|M| = 1$, $k = 1$, $|C| \geq 2$, the maximum degree of any vertex is $(r-1)$.*

Theorem 4.1. In a k-intersecting hypergraph with $|M| = 1$, $k = 1$, $|C| \geq 2$, the maximum number of edges is $(r-1)^2$.

The question remains as to how to generate a maximally sized hypergraph, and in what cases we may generate these hypergraphs. We find that tightness holds for prime powers of the form $r - 1 = p^d$.

Theorem 6.1. *If m gives the maximum number of edges in an r-partite 1-intersecting hypergraph, $m = (r-1)^2$ if and only if r is a prime power of the form p^d.*

We find an equivalence between maximizing the 1-intersecting hypergraphs and maximizing a set of mutually orthogonal Latin squares [7]. A *Latin square* is an $n \times n$ matrix in which n distinct symbols from a symbol set S are arranged, such that each symbol occurs exactly once in each row and in each column. A *reduced Latin square* is a Latin square in which the first row is in the natural order of the symbol set we choose. In this paper, let the ordered symbols of an order $(r-1)$ Latin square be $\{1, 2, \ldots, (r-1)\}$. Two Latin squares L_1 and L_2 are said to be *orthogonal* if $\exists$ for each ordered pair $(i, j) \in \{1, 2, \ldots, (r-1)\} \times \{1, 2, \ldots, (r-1)\}$ exactly one choice of row k and column l such that $L_1(k, l) = i$ and $L_2(k, l) = j$. A set of Latin squares $L_1, L_2, \ldots, L_k$ is mutually orthogonal if L_i and L_j are orthogonal for all pairs.

The following definitions and theorems on MOLS and 1-intersecting hypergraphs allow us to characterize when construction of maximally sized 1-intersecting hypergraphs is possibly and implicate one of many possible constructions.

We introduce the terminology of a *bundle* in a 1-intersecting hypergraph. A bundle in a 1-intersecting hypergraph is a set of $(r-1)$ edges incident on some vertex of maximum degree in V_1. WLOG, we assign the edges in the first bundle as the symbols [1 1 1 ... 1][2 2 2 ... 2]... [$(r-1)$ $(r-1)$ $(r-1)$ ... $(r-1)$]. For all 1-intersecting bundles besides the first, each edge may be represented as a permutation of the numbers $1...(r-1)$. Therefore, all bundles besides the first take the form of Latin squares. Two bundles v_{1i} and $v_{1i'}$ are said to be 1-*intersecting bundles* if $\forall$ rows in v_{1i} indexed by k and $\forall$ row $\in v_{1i'}$ indexed by l, $\exists$ some j st $v_{1i}(k, j) = v_{1i'}(l, j)$. We conjecture and prove the following equivalence.

Theorem 6. *The problem of maximizing the number of 1-intersecting bundles is equivalent to that of finding (r-2) mutually orthogonal Latin squares.*

We show this by way of the below theorems and the corollaries which follow. We prove equivalence between the problem of finding mutually orthogonal Latin squares and 1-intersecting bundles defining a bijective transformation T on a Latin square M from a set of mutually orthogonal Latin squares to another Latin square A which we consider as a bundle.

$$T : M \to A := \{A_{kj} = i | M_{ij} = k\} \quad [7]$$

Theorem 7 *If M and N are two reduced mutually orthogonal Latin squares, then $A = T(M)$ and $B = T(N)$ are two 1-intersecting bundles.*

Theorem 8 *If M and N are two reduced mutually orthogonal Latin squares, then $A = T(M)$ and $B = T(N)$ are two 1-intersecting bundles with intersections defined by the 1's on their diagonals.*

Theorem 8.1. *The maximum number of MOLS of order q is $q - 1$.* [8]

Theorem 9. *Finding a set of $(q - 1)$ MOLS is possible iff $(q - 1)$ is a prime power of the form $q = p^d$.* [8]

Generating $(q - 1)$MOLS is typically achieved using three different constructions. The general form for generating $(q - 1)$ MOLS is a finite field construction. Others of note are the Bose construction and the direct product construction [9]. We pose a novel construction for prime powers of the form $q = p^d$ such that $pd + 1$ is a prime power and the set of permutations $\{I, C_p, C_p^{-1}\}$ is strictly transitive, where C_p is a cyclic permutation on the symbols $1..p$. We call this approach the cyclic permutation approach. We provide an algorithm which generates an ordered set of permutations Π and a permutation σ that have the following properties.

- For two $\pi_i, \pi_j \in \Pi$, $\pi_i(k) \neq \pi_j(k) \forall\ k$.
- The element k takes on the index j in the ith row given by the ith element in the kth column; that is, $\pi_i(j) = k | A_{ik} = j$.
- The diagonal elements are σ^2.
- All π_i can be generated by some combination of other π_js.
- All $\pi_i = \pi_i^{-1}$

Our algorithm generates up to $(r-2)$ 1-intersecting bundles which are given by the set $A = \{\Pi, \Pi\sigma, \Pi\sigma^2 \ldots \Pi\sigma^{(r-2)}\}$. If a permutation σ has one fixed element and one $(r - 2)$ sized orbital, then for every $(a, b) \in (2...(r - 1))^2$, $\exists$ a unique σ_i for which $\sigma_i(a) = b$. We proceed under the assumption that the properties above hold, and we use this to prove the following:

Corollary 7.2 If the cyclic permutation approach generates a set of permutations Π such that $\pi_i^{-1} = \pi_i$ and $\pi_i\pi_i = I$; and for two $\pi_i, \pi_j \in \Pi$, $pi_i(k) \neq pi_j(k) \forall\ k$; and a permutation σ such that the element 1 is fixed and the subsequent $r-2$ elements form an $(r-2)$ orbital, then the bundles defined as $A = \{\Pi, \Pi\sigma, \Pi\sigma^2 \ldots \Pi\sigma^{(r-2)}\}$ will be $(r - 2)$ valid 1-intersecting bundles.

We prove this via a series of bijective transformations to create a set of $(r - 2)$ mutually orthogonal Latin squares, first casting the bundles to MOLS and then casting MOLS to reduced MOLS. We guarantee $(r-2)$ 1-intersecting bundles under these conditions. Furthermore, we may generate $(r-2)$ *symmetric* 1-intersecting bundles using the cyclic permutation approach and a set of bijective transformations we define. A Latin square L of order n is *symmetric* if $L(i, j) = L(j, i)$ for all $0 \leq i, j \leq n - 1$. Casting the original $(r - 2)$ 1-intersecting bundles to MOLS, placing the resulting identity MOLS in reduced form, and finally transforming reduced form MOLS back to bundles, we show that it is possible to generate a set of $(r - 2)$ MOLS and $(r - 1)$ symmetric 1-intersecting bundles using the cyclic permutation approach. We make use of the

important property that for a set of MOLS, a random permutation on the alphabet of the Latin squares does not affect the orthogonality of those Latin squares. [10].

We show that the construction guarantees certain properties for the different sets:

1. **Theorem 10.** The first row of all M is the permutation σ^{-1}.
2. **Theorem 11.** M^1, the transform $T^{-1}\Pi$, is symmetric.
3. **Corollary 11.1.** M^1's first column is σ^{-1}.
4. **Lemma 12.** M^1 has 1's along its diagonal.
5. **Theorem 14.**The first element of $M_{reduced}$, $M^1_{reduced}$, is a symmetric reduced Latin square.
6. **Theorem 15.** The ith row in $M^N_{reduced}$ is given by π_j where $j = \sigma^{-N}(i)$.
7. **Theorem 18.1.** For any $M^N_{reduced}$ the first element in a row i (which is equivalent to the column) gives the positions of the element 1 in the row i; that is, $M^N_{reduced}(i,1) = k \implies M^N_{reduced}(i,k) = 1$.
8. **Theorem 19.**If $M^1_{reduced}$ is a reduced form Latin square, $A^1_{reduced} = T(M^1_{reduced}) = M^1_{reduced}$.
9. **Theorem 19.1.** $A^N_{reduced} = T(M^N_{reduced})$ for any $M^N_{reduced} \in M_{reduced} \implies A^N(i,i) = 1$.

The combination of Theorems For a complete set of symmetric bundles, we transform $A_1 \dots A_{r-2}$ to $M_1 \dots M_{r-2}$, then normalize the MOLS symbols in M_1, and then cast back to bundles, the bundles are symmetric with first row and column both given by a power of σ.

We show the cyclic permutation succeeds for certain prime powers. We are left with the open question of defining for all prime powers when the cyclic permutation approach applies, and conjecture that it is so when a prime power p^d has $pd+1$ is a prime power which also takes the same form.

Conjecture 22. *If $(r-1)$ is a power of a prime p, expressed $(r-1) = p^d$, such that $pd+1$ is prime and has the same property, then the result returned by the cyclic permutation algorithm Π and σ have the property that all $\pi_i^p = I$, and the element 1 is fixed in σ with the subsequent $r-2$ elements forming an $(r-2)$ orbital.*

Further work is needed to show this result, but in fact if possible this could have implications for generating large order MOLS in much smaller time and space than required of the construction using $GF(r-1)$ [11] or the direct product construction using Latin sub-squares [12].

2 Background

We begin with some general motivation on vertex cover in graphs and hypergraphs, and the inspiration for our polynomial-time approach. We then pose the question of hypergraphs which are intersecting; that is, the maximal matching size is 1.

For orders of hypergraph larger than three, it is known that the questions of finding a maximal matching or a minimum vertex cover are NP-hard. Thus there is thought to be no polynomial time algorithm for finding the maximal matching or the minimal vertex cover. A hypergraph that is not r-parite is said to be unpartitioned. In an unpartitioned hypergraph, Bansal and Khot show that it is impossible to find an approximation algorithm for vertex cover that executes in polynomial time even when the hypergraph is close to r-partite [13]. In the r-partite r-regular case, Guruswami, Venkatesan and Sachdeva [4] show NP-hardness of obtaining an approximation factor of $(\frac{r}{4} - \epsilon)$ for even r and $(\frac{r}{4} - \frac{1}{4r} - \epsilon)$ for odd r, NP-hardness of obtaining a nearly optimal approximation factor of $(\frac{r}{2} - 1 + \frac{1}{2r} - \epsilon)$, and an optimal unique games-hardness

for approximation within factor $(\frac{r}{2} - \epsilon)$, thereby showing the optimality of Lovász's algorithm if one assumes the Unique Games conjecture. Ron Aharoni showed in 2001 [6] that in any tripartite hypergraph, $|C| \leq 2|M|$ using a topological proof, but does Aharoni does not pose an algorithm for the vertex cover. Therefore, the question of finding a polynomial time vertex cover algorithm in the general r-partite r-uniform hypergraph is not conceivable and we must further constrain the setting.

In the case where the maximal matching of the hypergraph is 1, the hypergraph is said to be *intersecting*. A famous theorem by Erdös-Ko-Rado bounds the size of the intersecting hypergraph for a uniform intersection of size k; that is, all edges intersect all other edges exactly k times.

Theorem 1. *(Erdös-Ko-Rado). Let $k \leq n/2$ and G be a k-uniform, k-intersecting hypergraph on vertex set V. Then $|G| \leq \binom{|V|-1}{k-1}$. Furthermore, $|G| = \binom{|V|-1}{k-1} \iff \exists v \in V$ st $G = \{e \in \binom{|V|}{k} : v \in e\}$. [14].*

The size of an intersecting hypergraph is bounded. Bohman and Martin bound the size of sparse hypergraphs using the idea of clique number [15]. They use the clique number w to show a bound on the k-intersecting hypergraph of $|G| \leq \binom{n}{k} - \binom{n-w}{k}$. In addition, they show that if G is a k-intersecting family of maximum cardinality then there exists a maximum clique K in G such that G contains all k-sets that intersect K [15]. In fact it is not known if there exists any polynomial time algorithm for vertex cover in the intersecting hypergraph, and in fact even in the intersection 1 case it has been shown that intersecting Set Cover problem cannot be approximated within a $o(logn)$ factor in random polynomial time unless $NP \in ZTIME(nO(\log\log(n)))$ [16].

Approximation algorithms for an r-partite hypergraph have been proposed by Halperin in [17], who finds that for a partitioned hypergraph, there is an almost trivial r-approximation by finding a maximal matching, and the best algorithms achieve only a slight improvement of $r(1-o(1))$ over this approximation. In a hypergraph which is close to r-partite, finding vertex cover is inapproximable in less than NP time. They conclude that for every $r \geq 2$, the problem is inapproximable within $r - \epsilon$ even when the hypergraph is almost r-partite [17].

A related conjecture to Ryser's Conjecture is Tuza's Conjecture, which concerns a 3-uniform unpartitioned graph. If G is a graph, and H is the 3-uniform hypergraph whose vertices are the edges of G and whose edges are the sets of three edges in G that form $3-$cycles, then the conjecture states that if G has at most r edge-disjoint triangles, deleting some set of $2r$ edges breaks all triangles [18]. Although these hypergraphs are 3-uniform, they are not 3-partite, so this requests the same bound as in Ryser's and Jones' Conjectures, but for a different family of hypergraphs. The conjecture has been proved when G is planar [18], has been exhaustively proved for order ≤ 7 by Puleo [19], and proved when it is tripartite [20]. Krivelevich also proved that the inequality holds when either parameter is replaced with its fractional version [21]. We next briefly touch on the *fractional* hypergraph.

Other interesting work poses the idea of fractional covers and matchings. A fractional cover of $G = (V, E)$ is a weighting $a : V \to \mathbb{R}^+$ so that $\sum_{x \in S} a(x) \geq 1$ for every $S \in E$, and the weight of this cover is $\sum_{x \in V} a(x)$. By a fractional matching is an edge-weighting $b : E \to \mathbb{R}^+$ so that $\sum_{S \ni x} b(S) \leq 1$ for every $x \in V$, and the weight of this matching is $\sum_{S \in E} b(S)$. We define ν^* as the minimum of the fractional cover and τ^* as the maximum of the fractional matching. By LP-duality, $\nu^* = \tau^*$ is satisfied for every hypergraph. For r-regular r-partite hypergraphs, $\tau^* \leq (r-1)\nu$ and $\tau \leq \frac{1}{2} r \nu^*$ [2].

The denominating factor across all of these domains is that the question of finding a polynomial time *minimum* vertex cover algorithm in polynomial time for the general r-partite r-uniform hypergraph is not conceivable. We must further constrain the setting and possibly turn to approximation to achieve results. The *intersecting* hypergraph is one way we can limit the hypergraph size.

In the case where the maximal matching of the hypergraph is 1, the hypergraph is said to be *intersecting*. A famous theorem by Erdös-Ko-Rado bounds the size of the intersecting hypergraph for a uniform intersection

of size k; that is, all edges intersect all other edges exactly k times.

Theorem 1 (Erdos-Ko-Rado). *Let $k \leq n/2$ and G be a k-uniform, k-intersecting hypergraph on vertex set V. Then $|G| \leq \binom{|V|-1}{k-1}$. Furthermore, $|G| = \binom{|V|-1}{k-1} \iff \exists v \in V$ st $G = \{e \in \binom{|V|}{k} : v \in e\}$ [14].*

The size of an intersecting hypergraph is bounded. As explored above, Bohman and Martin bound the size of sparse hypergraphs using the idea of clique number [**?**], using a clique number w to bound on the k-intersecting hypergraph of $|G| \leq \binom{n}{k} - \binom{n-w}{k}$ [**?**].

In fact work concerning intersecting hypergraphs may hold the key to general hypergraphs. We consider a subgraph G' to be a general hypergraph's *intersection graph*: Given a system G of n sets, their intersection graph is a graph G_S whose vertices are the elements of G and vertices v_1, v_2 being connected by an edge if and only if $v_1 \cap v_2 \neq \emptyset$ [22].The work of Barry Guiduli and Zoltan Kiraly concerns finding the 'best' points for a paritioned intersecting hypergraph, and in particular quantifying how good the best selection of a single point or two points may be. Erdos and Gyhrfils (1990) considered k-intersecting problem for the best t points [14], and their work extends this with more concrete bounds. Weighting functions were used to quantify and prove the best choice of point(s) in these settings. In general, it was found that $L(k,n) := min(\frac{2}{k-1}, \frac{k(2n-k-1)}{n(n-1)})$, and tightness was only possible when $\exists$ a projective plane of order q [**?**]. We use the intuition of these approximations and findings to optimize the choice of edges and vertex cover in the reactive hypergraph.

The applications and relevant literature to this field is quite extensive, so we provide only a small window into algorithmic approaches to these different constrained problems. We now present the conjecture and domain of interest for the remainder of this paper.

3 Ryser's Conjecture for r-Partite Hypergraphs

3.1 Definitions

In mathematics, a graph (or hypergraph) $G = (V, E)$ is a set of vertices, V, and hyperedges, E. A vertex $v \in V$ is a node in the graph (or hypergraph), and an edge $e \in E$ can be considered a subset of vertices.

Let a hypergraph be designated $G = (V, E)$. We use the following definitions and notation for the remainder of the document.

Uniform. G is said to be *uniform* if $\forall e \in E$, $|e| = k$ for some constant k.

Partition. A partition is a grouping of all vertices $v \in V$ may be partitioned into groups $\{V_1, V_2 ... V_r\}$ such that $V = V_1 \cup V_2 \cup ... \cup V_r\}$, $V_i \cup V_j = 0$ if $i \neq j$.

R-partite. An r-partite hypergraph is a uniform hypergraph with r parts such that every edge $e \in E$ contains exactly one element from every part V_i, that is, $|e \cap V_i| = 1, \ \forall \ V_i$.

Let a r-partite hypegraph be $G = (V, E)$. Each edge $e \in E$ may be represented as $[v_{1j_1}, v_{2j_2}, ..., v_{ij_i}, ...v_{rj_r}]$. By r-partiteness, no edge contains more than one vertex in the same partition V_i, $i = 1...r$. Note that this is the same as saying one vertex from each partition i must be included in every edge.

Uniform r-partite hypergraphs have that $\forall e \in E$, $|e| = r$. In this document, G is assumed to be a uniform r-partite hypergraph unless otherwise specified. An edge refers to an r-edge in an r-uniform r-partite hypergraph. For example, for a tripartite hypergraph, $r = 3$, each 3-edge would be represented by 3 vertices

$[v_{i_1 j_1}, v_{i_2 j_2}, v_{i_3 j_3}]$.

In this document, we enumerate parts V_i st $\cup V_i = V$ and $V_i \cap V_i' = \emptyset$, $\forall i \neq i'$. We use the notation v_{ij} for a vertex which belongs to a part i with enumeration j.

Incident. A vertex $v \in V$ and an edge $e \in E$ of any hypergraph are said to be incident if $v \in e$. Then e contains v: $e = [v_{1j_1}, ..., v, ..., v_{rj_r}]$.

Degree. The *degree* of any vertex, v is d_v is size of its incident edge set. Then $d_v = |\{e \in E \ | v \in e\}|$.

Maximum degree. The maximum degree d_{max} is $max(d_v)$ for all $v \in V$.

Regular. A regular graph (or hypergraph) is one where all vertices have the same degree d_v. Then $|d_v| = k \forall v \in V$, for some constant k. A regular graph (or hypergraph) has maximum degree $d_{max} = k$.

Matching. A *matching* M is a selection of r-edges $M = \{ \{e_i\} \mid e_i = [v_{1j_1}, ..., v, ..., v_{rj_r}]\}$ such that no two edges $\{e_1, e_2 .... e_{|M|}\}$ share a vertex.

Maximum matching. A maximum matching of an r-partite hypergraph is any matching with the maximal cardinality. That is, the size of the set of r-edges is the largest possible subset of E such that no two edges share a vertex.

There may be more than one maximum matching. We denote the cardinality of a maximum matching $M^\star$ as $|M|$. $|M|$ is the same for all maximum matchings.

Vertex cover. A vertex cover C of a graph (or hypergraph) is a selection of vertices $\{v\}$ st all edges in the graph (or hypergraph) include at least one vertex $v \in C$. That is, $\forall e \in E$, $\exists\ v_i \in C$ st $v_i \in e$.

Minimum vertex cover. A minimum vertex cover is any vertex cover with the minimal cardinality. We denote such a vertex cover as $C^\star$ and the cardinality of its matching $|C|$. $|C|$ is the same for all minimum vertex covers of an r-partite hypergraph.

Intersecting. Two edges e and f are said to be *intersecting* if $|e \cap f| \geq 1$.

Linear. A hypergraph is said to be *linear* if any two vertices exist in at most one edge. For edges e and $f \in E$, $e \neq f$, $|e \cap f| \leq 1$.

k-intersecting. Two edges e and f are said to be *k-intersecting* if $|e \cap f| = k$ for some constant k.

k-intersecting hypergraph. A hypergraph is called a *k-intersecting hypergraph* if all its edges are k-intersecting; that is, all edges intersect all other edges at exactly k points; $\forall$ edges e and $f \in E$, $e \neq f$, $|e \cap f| = k$.

1**-intersecting hypergraph** A 1-intersecting hypergraph is the class of hypergraph which is both linear and intersecting. $\forall$ edges e and $f \in E$, $e \neq f$, $|e \cap f| = 1$.

An illustration of an r-partite r-uniform hypergraph follows. Each edge takes the form $[v_{1j_1}, ..., v_{ij_i}, ..., v_{rj_r}]$ and each vertex is said to be in a part V_i.

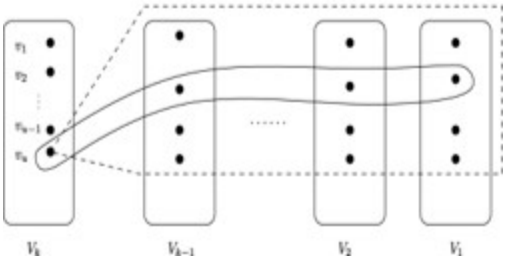


Figure 1: Generalized r-partite hypergraph [23]

3.2 Ryser's Conjecture

Let $G = (V, E)$ be an r-partite r-uniform hypergraph. Let the maximum matching of G be M with $|M|$ edges. Let the minimum vertex cover of G be C with $|C|$ vertices.

Conjecture 2. ***Ryser's Conjecture.*** *For an r-partite r-uniform hypergraph,*

$$| C | \leq (r-1) | M | \quad [3].$$

Ryser's Conjecture is an open question for intersecting hypergraphs, and in fact it is open even for linear intersecting hypergraphs. The following sections first consider the intersecting case and then the linear intersecting hypergraph.

4 Properties of an Intersecting Hypergraph

The class of hypergraphs for which $|M| = 1$ is the class of hypergraphs which are *intersecting*. This is the set of hypergraphs for which every edge shares at least one vertex with every other edge; any two edges intersect at least once.

When applying Ryser's conjecture to this type of hypergraph we have that $|C| \leq (r-1)|M|$, $|M| = 1$ implies

$$|C| \leq (r-1).$$

Thus it should always be possible to find a vertex cover for any r-partite hypergraph using only $(r-1)$ vertices. We pose a candidate algorithm and framework to verify this conjecture.

4.1 Empirical findings

We take an algorithmic approach to the $|M| = 1$ case for the r-partite hypergraph. It is centered around the key observation that if there exists ahypergraph with $|M| = 1$ for which no choice of $(r-1)$ vertices specifies a valid vertex cover, Ryser would be debunked. If, however, no suchhypergraph exists, then the conjecture is still valid. For small r, we empirically show that all possible graphs have vertex covers of cardinality at most $(r-1)$.

The choice of a vertex cover is not trivial; in fact, finding a vertex cover in a hypergraph is NP-hard. However, in the $|M| = 1$ case, we pose a vertex cover algorithm for an $(r-1)$ sized cover which achieves $O(r(V+E))$ amortized time.

We propose an approach to the minimum-vertex-cover/ maximum-matching problem which generates an $(r-1)$-sized vertex cover for anyhypergraph with a maximal matching of size 1. Our algorithm succeeds empirically on all possible hypergraphs with $r \leq 5$. To handler larger order graphs, we define an *optimal reactive hypergraph*, a reactive hypergraph framework which attempts to maximize hypergraph size in response

to a vertex cover algorithm. Our vertex cover algorithm succeeds in response to optimal hypergraphs with r up to 20. The reactive hypergraph back-and-forth framework and the results on our candidate algorithm are elaborated below.

4.2 Reactive hypergraph approach

To establish a baseline for this algorithmic approach, consider a game consisting of two players, Alice and Bob. One of the players, Alice, is convinced that as long as the size of the maximal matching of the hypergraph is 1, she can generate a $(r-1)$ sized vertex cover. Bob is convinced that he can play an edge such that Alice's choice of $(r-1)$ vertices is not a valid vertex cover.

If Alice wins, her algorithm is a valid candidate vertex cover algorithm. If Bob wins, her algorithm is debunked. The play-by-play of the game proceeds as follows.

- Begin with an empty r-partite hypergraph. Since no vertices exist in the hypergraph, it is easy to see that $C = \emptyset$.
- WLOG, add an arbitrary edge to the hypergraph.
- Alice generates a vertex cover as per her vertex cover algorithm.
- Bob adds an edge to the hypergraph as per his edge choice algorithm (brute force or greedy).
- Continue until no more possible edges may be added.

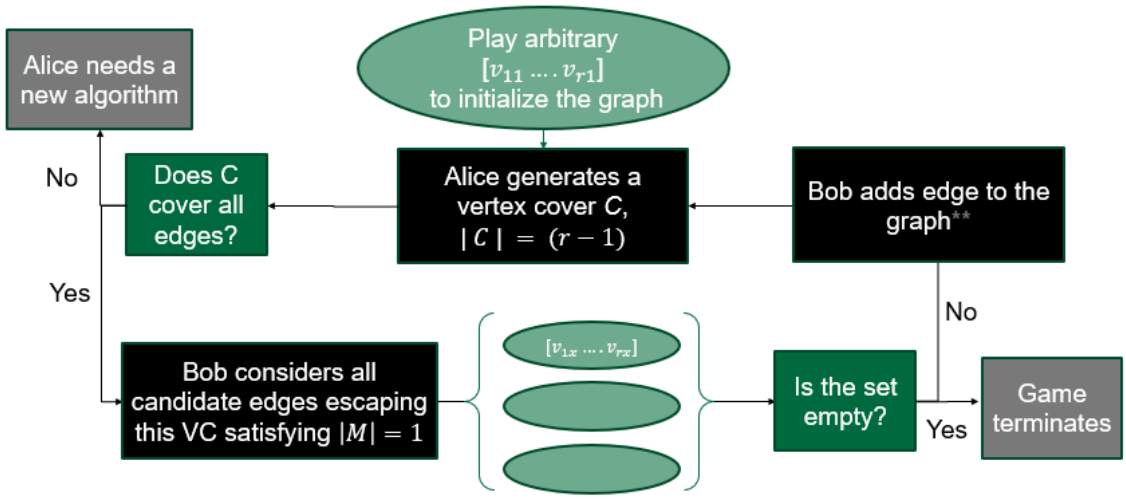


Figure 2: Outline of the Alice/Bob reactive hypergraph game

If the above back-and-forth play does not terminate, then we have found a hypergraph configuration for which the vertex cover algorithm does not find an appropriate $(r-1)$ sized vertex cover for some hypergraph with a maximum matching size 1.

We must define two algorithms to play the game: 1) Alice's choice of vertex cover and 2) Bob's choice of edge. The sections below explore these two algorithms.

4.3 Vertex cover algorithm

As outlined, Alice is convinced that as long as the size of the maximal matching of the hypergraph is 1, she can generate a $(r-1)$ sized vertex cover. Notice that a brute-force approach to an algorithm for vertex cover would be to consider all $(r-1)$ sized subsets of vertices, then check every edge to see if all edges are hit. However, this is a combinatorial approach and thus highly inefficient, executing in $\binom{O((n}{(r-1)))=O(n!)}$ time. We propose an algorithm which chooses a candidate vertex cover according to edge participation in a hypergraph.

Alice constructs a vertex cover as follows.

- Initialize the solution, $C_{op} = \{\}$
- Initialize the set $E_{remaining} = E$ to be all current edges in the hypergraph.
- Recurse until $|C_{op}| = (r-1)$ or $E_{remaining} = \emptyset$:
  - Consider the edges $E_{remaining}$ which have yet to be hit by the current vertex cover.
  - Select the vertex v with maximum participation in $E_{remaining}$; i.e. the vertex with the maximum number of incident edges in $E_{remaining}$. Break ties by part and vertex number; i.e. $V_1 > V_2 > V_3$... and $v_{11} > v_{12} > v_{13}$....
  - Add v to the cover and remove all incident from the set $E_{remaining}$. Let the function $incident(v, E_{remaining})$ return the set of edges in $E_{remaining}$ incident on v. $E_{remaining} = E_{remaining} - incident(v, E_{remaining})$.
- If $|C_{op}| \leq (r-1)$, fill in the remaining vertices in their natural ordering; $V_1 > V_2 > V_3$... and $v_{11} > v_{12} > v_{13}$....
- Return C_{op}, a vertex cover of size $(r-1)$.

Assuming hashmaps of vertices to incident edges, this algorithm proceeds in $O((r-1)(V+E))$ time. There are $(r-1)$ vertices to be selected, (r assumed a small constant). It takes $O(V)$ time to select the optimal vertex and it takes $O(E)$ time to fix up the vertex mapping after selecting said vertex, in the worst case. Each edge may only contribute r, a small constant, time in fixing up the vertex mapping after it has been determined to be incident. Therefore, the algorithm executes in $O(r(V+E))$ time and uses $O(rVE)$ space to store the mappings.

4.4 Edge choice algorithms

To prove to effectiveness of the algorithm on any r-partite hypergraph, we pose the reactive hypergraph framework, which responds to a choice of vertex cover with a new edge which escapes it at each given time. We start with an arbitrary edge and add edges to the hypergraph one at a time. Edges are chosen in reaction to the current choice of vertex cover and all the existing previous edges. We take 2 approaches to characterizing the choice of edge in reactive hypergraphs:

- Brute-force approach. Given a hypergraph and a choice of vertex cover, explore all possible additions of edges (WLOG), starting off a new 'game' for each possible edge choice.
- Optimal approach. Given a hypergraph and a choice of vertex cover at a given time step, attempt to play a new edge 'optimally' in reaction; that is, play an edge which will allow the maximum number of edges to be played.

The two options are explored and characterized in Sections 4.4.1 and 4.4.2.

4.4.1 Brute-force edge choices

The brute-force approach to building out a hypergraph is defined as recursively exploring every possible edge to add at every given time step with the same vertex cover algorithm applied to the hypergraph at each timestep to drive the choices.

The brute-force algorithmic approach considers all possible choices of new edges, recursively building all possible hypergraphs. At each time step, it generates the set of possible edges given the posed vertex cover and the state of the hypergraph. All possible edges must escape the vertex cover, but still fall within the constraints $|M| = 1$ and r-partiteness. It then tries out each of Bob's possible edge choices in a depth-first manner. A branch of the recursive algorithm terminates when there are no more edges to play. When a game terminates, the algorithm traverses back up the tree, removing edges until there are other options and recursively exploring those.

If any hypergraph built by this algorithm does not terminate, then the vertex cover algorithm can be considered debunked. However, the number of possible hypergraphs increases exponentially in r, and the program may not terminate empirically for this reason. As long as each game terminates within reasonable time, we cannot invalidate a posed vertex cover algorithm.

The choice of edges at each time step may be thought of as a tree. As a toy example, consider the $r = 3$ case. We now prove that $|C| \leq 2$ for an intersecting hypergraph where $r = 3$.

Theorem 3. *If $G = (V, E)$ be a tripartite hypergraph with $|M| = 1$, then $\mid C \mid \leq 2$. That is, for some $(x, y) \in V$, every 3-edge contains either x, or y, or both.*

Proof. We know that the hypergraph necessarily contains some edge, since M=1. WLOG, call this first edge (v_{11}, v_{21}, v_{31}), as per the convention. All 3-edges $\in E$ must be of the form $(v_{1i}, v_{2j}, v_{3k}$. Otherwise they would not be valid edges in the tripartite hypergraph. Then we know that any edge introduced in the hypergraph $(v_{1i}, v_{2j}, v_{3k}$ must necessarily share some vertex with every other edge already in the hypergraph. Otherwise, a matching M could be selected with cardinality 2. WLOG, let this vertex be v_{11}. Then edges include $[v_{11}, v_{21}, v_{31}]$ and $[v_{11}, v_{22}, v_{32}]$. Now we must add some edge which intersects both these edges. We may either play v_{11} and then either play an existing vertex or generate a new one for the remaining slots, or play v_{21} and v_{32} and fill the remaining slot, or play v_{22} and v_{31} and fill the remaining slot. Regardless of the path we choose, if we do not play v_{11}, in the next edge we will not be able to generate any new vertices. If we play v_{11}, we might play $v_1 1$ infinitely; but then we would still have $|C| \leq 2$.

To help with this ambiguity, consider negating the above. The claim is equal to the statement "For any selection of $(x, y) \in V$, $\exists$ some 3-edge $\in E$ containing neither x nor y." That is, given any pair of vertices, we can find an edge escaping it. This is where the reactive hypergraph comes into play. We proceed by building up the hypergraph gradually given selections of vertices $\in V$ which have been included in some edge $\in E$.

Because of the lengthiness of the proof above, the reactive hypergraph narrows down cases where the choices may be infinite (i.e. where we play infinitely on a single vertex and play randomly for the remaining vertices). The tripartite case is deterministic. We prove that the algorithm performs correctly in Lemma 3.1

Lemma 3.1. *The tripartite reactive hypergraph with $|M| = 1$ deterministically terminates under the posed vertex cover algorithm.*

Proof. At the beginning of the back-and-forth, the edge $[v_{11}, v_{21}, v_{31}]$ is initialized. There must be at least one edge in the hypergraph, since M=1. The counts are now $\{v_{11} = 1, v_{21} = 1, v_{31} = 1\}$. Alice chooses her vertex cover as per the algorithm: (v_{11}, v_{21}). Bob now *must* play $[v_{12}, v_{22}, v_{31}]$, because he must have at least one edge in common with all previous edges. As he cannot choose v_{11} or v_{21}, he must choose v_{31}. Then he must create two new vertices in V_1 and V_2: v_{12} and v_{22}, respectively. The vertex counts are now

$\{v_{31} = 2, v_{11} = 1, v_{12} = 1, v_{21} = 1, v_{22} = 1\}$. Alice now chooses (v_{31}, v_{11}). As she has selected two vertices from the first edge, the new edge must necessarily include v_{21}. Then, as it cannot include v_{31}, the only remaining option to be included from 2) is v_{12}. As v_{31} cannot be included, we generate a new v_{32}. So Bob must necessarily play the edge $[v_{12}, v_{21}, v_{32}]$. Counts are now $\{v_{12} = 2, v_{21} = 2, v_{31} = 2, v_{11} = 1, v_{22} = 1, v_{32} = 1\}$. Alice selects (v_{12}, v_{11}). Bob now must play a vertex in V_1 but cannot play v_{11} or v_{12}; he therefore plays v_{13}. He then might choose v_{21} or v_{22} for his vertex in V_2, but if he chooses v_{22} he hsa no chance of hitting both $[v_{11}, v_{21}, v_{31}]$ and $[v_{12}, v_{21}, v_{32}]$ in his choice of vertex in V_3. Therefore he must play $[v_{13}, v_{21}, v_{32}]$. Finally Alice chooses (v_{21}, v_{12}) at which point Bob has no possible play and the hypergraph terminates. □

□

Given Alice's vertex cover algorithm these are the only sequence of plays that Bob can make, which is why we are able to provide the complete proof in-line. For higher orders of r, Bob has many choices at any given step. We show for $r = 4$ that there are 260 possible hypergraphs, all of which terminate. Transcripts for some maximum sized hypergraphs are shown in Figure 5 and 7.
We brute-force all possible hypergraphs for $r = 3$ and $r = 4$ in Java to verify the result. However, the number of possible reactive hypergraphs grows quickly. In Table 1, the number of possible hypergraphs (i.e. the number of terminated 'games' using the back-and-forth algorithm) is outlined. The recursion becomes astronomically large even for low orders of r. Even in the $r = 5$ case, the number of hypergraphs generated is over 123000. Thus, other algorithmic options must be considered. In the table below, 'maximum size' refers to maximum observed number of edges for one of the hypergraphs.

r	# of reactive hypergraphs	maximum size
3	1	4
4	260	13
5	> 123000	≥ 34

Table 1: Number of possible brute-force hypergraphs and maximum achieved size

For $r = 5$, the algorithm was run continuously over four days. 123000 possible hypergraphs were explored. All hypergraphs terminated within between .59 seconds and 48 seconds, with the maximum time dependent on the number of edges. Memory bloat also likely contributed to running time. However, considering that the rate was consistently bounded, we cannot conclude that Bob was ever able to escape Alice, even though the depth-first-search did not terminate. For this reason, we anticipate that all hypergraphs *would* terminate, though we are unable to show so using brute-force. We turn to other methods for validation.

4.4.2 Greedy edge choice

A greedy edge choice attempts to maximize its intersection with the edges which already exist in the hypergraph. This choice is added to the hypergraph in direct response to the choice of vertex cover and the current state of the hypergraph. The intuition is that a hypergraph can achieve maximal number of edges by adhering to a play-style which favors using previously played edges in the new edge selection.

Consider G_r to be a reactive greedy hypergraph; that is, edges may only be added by considering the greedy vertex cover and all the existing edges and in accordance to r-partiteness. The greedy edge choice is defined as follows: For each vertex cover at each time step, explore all maximally participating vertices which escape the cover. The edge choice is similar to Alice's choice of vertices, but an edge may not choose more than one vertex in the same part, may not choose a vertex which is in the current cover, and in fact must select r vertices, not $(r - 1)$. The algorithm is as follows, given a cover C and a hypergraph $G = (V, E)$.

- Initialize an empty edge, given by an r-sized array which will be filled with vertices from $V_1 ... V_r$ for each $i = 1...r$th element in the array. $e = [\ ]$
- Initialize the mapping $V_{remaining}$ which maps part number i to possible remaining vertices to indicate the parts which have yet to be set in the edge and the possible choices of vertex in these parts. Add all V to $V_{remaining}$, then remove the vertices in C.
- Initialize the set $E_{remaining} = E$ to be all current edges in the hypergraph.
- Recurse until e full, any part V_i has $V_{remaining}(i) = \emptyset$, or $E_{remaining} = \emptyset$:
  - Consider the edges $E_{remaining}$ which have yet to be hit by the current vertex cover.
  - Select the vertex v in the entries of $V_{remaining}$ with maximum participation in $E_{remaining}$; i.e. the vertex with the maximum number of incident edges in $E_{remaining}$. Break ties by part and vertex number; i.e. $V_1 > V_2 > V_3 ...$ and $v_{11} > v_{12} > v_{13} ....$
  - Add v to the edge and remove the part from parts which must be hit; $V_{remaining} \leftarrow V_{remaining} - V_{remaining}(i)$.
  - Also remove all incident edges from the set $E_{remaining}$. Let the function $incident(v, E_{remaining})$ return the set of edges in $E_{remaining}$ incident on v. $E_{remaining} = E_{remaining} - incident(v, E_{remaining})$.
- If $E_{remaining}$ is non-empty, Bob loses.
- If $V_{remaining}$ is non-empty, fill the edges using $V_1 > V_2 > V_3 ...$ and $v_{11} > v_{12} > v_{13} ....$
- Return e.

For one choice of greedy edge, we rely on an ordering $V_1 > V_2 > V_3 ...$ and enumeration $1 > 2 > 3 ....$ However, we really would like to to consider all possible greedy edges (choosing, at a given time, the vertices for the edge which have maximal participation in all the other previously played edges, ignoring the edges already hit by vertices chosen). This can also be done by removing the clause of the tie-breaking and recursing for all possible hypergraph greedy edge choices. While a slower-growing exponential than the brute-force approach which explores all hypergraphs, this is still quite slow.

We find that in the greedy reactive hypergraph for $r = 3..20$:

r	3	4	5	6	7	8	9	10
# edges in hypergraph	4	11	17	23	64	165	308	1181
r	11	12	13	14	15	16	17	18
	2288	1906	8638	421	926	1717	2270	3128
r	19	20						
	3392	6252						

Table 2: Size of greedy reactive hypergraph for r=3..20

Bob always loses for all $r = 3...20$ when he chooses edges per the greedy edge algorithm. Thus Alice is always able to find an $(r-1)$ sized vertex cover using her algorithm. The number of edges he is able to play in this case is show in Table 2. No known mathematical sequences are represented by these numbers. Therefore, we leave quantifying this play as an open route of exploration.

4.5 Maximizing depth

We can see above that the maximum brute-force depth for $r = 4, 5$ was not achieved by the greedy algorithm. Therefore, we pose the question of what the true optimal algorithm should be, to allow Bob to play for as

long as possible without hitting a vertex cover and a set of previously chosen edges such that he cannot pick any valid new edges.

To do this, we characterize the reactive hypergraph for $r = 4$. The maximum observed size of for this hypergraph was thirteen, whereas the greedy algorithm produced a hypergraph with 11 edges. In the brute-force reactive hypergraph, $r = 4$, 260 possible hypergraphs were explored. 3 of these hypergraphs achieve minimum depth and 32 achieve maximum depth. This section aims to explore what characteristics led to larger sized hypergraphs.

# edges	# occurrences
7	3
8	14
9	6
10	63
11	116
12	26
13	32
total	260

Table 3: Categorizing hypergraph size for the brute-force hypergraph

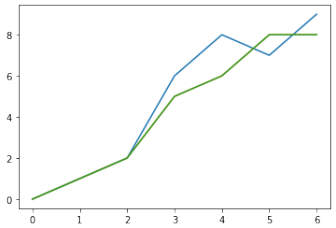
Transcripts of all 260 games were generated and analyzed using Python, Jupyter+Pandas. We can visualize the choices of edges Bob plays, in order to better understand what characterizes optimal play. Let m be the size of the hypergraph generated; $m \in \{7...13\}$. We characterize the 'score' of a game as the number of edges a choice of edge hits at a given point. Then the edge choices are enumerated by their scores below for $m = 7...13$.

There are several key observations to be made here.

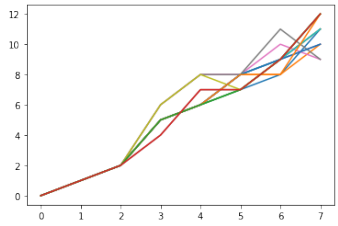
- Bob does not play greedily in any of the 32 cases where he generates the max-size hypergraph.
- Any max-size hypergraph (size 13) always ends with the vertex cover $[v_{21}, v_{22}, v_{13}$. This is interesting because the vertex cover does not consist of vertices of all the same part.
- The max-depth hypergraphs' second-to-last choice of vertex cover always begin with v_{41}. They then either select another vertex in V_4 or a vertex in V_1. The third step is always a vertex in V_1 (this makes sense because it employs Alice's tie-breaking trait).

The transcripts of all maximum hypergraphs are included in the GitHub repo in the file "MaxDepthTranscriptsR=4."

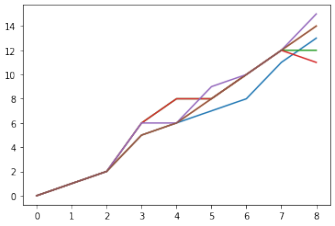
Figure 3: Visualizing edge hits for hypergraph sizes $m = 6...13$.



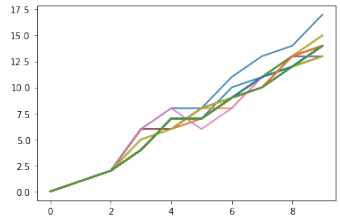
(a) m=7



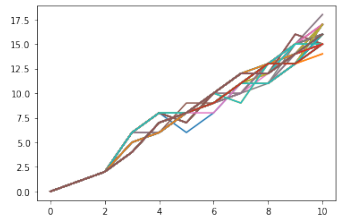
(b) m=8



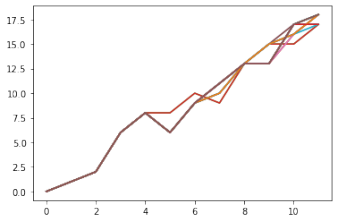
(c) m=9



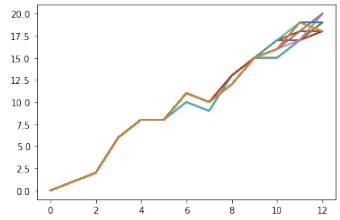
(d) m=10



(e) m=11

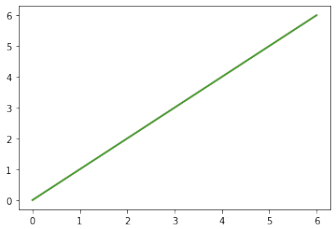


(f) m=12

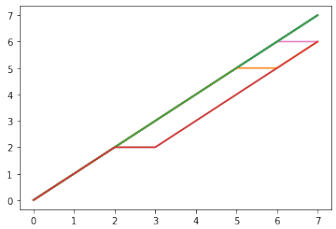


(g) m=13

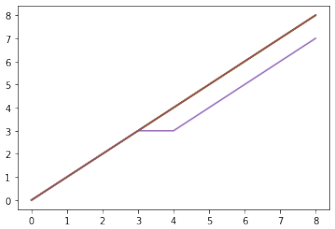
Figure 4: Visualizing greediness for hypergraph sizes $m = 6...13$.



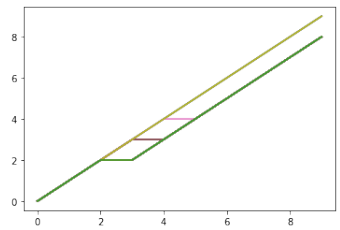
(a) m=7



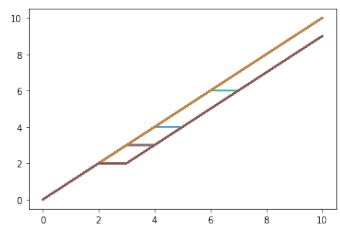
(b) m=8



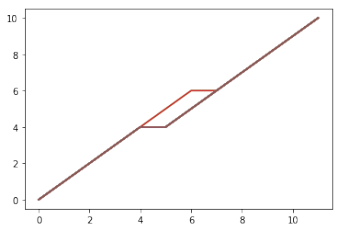
(c) m=9



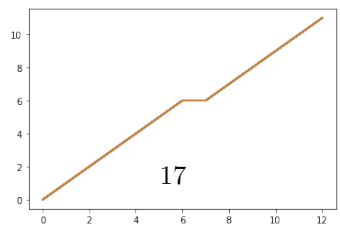
(d) m=10



(e) m=11



(f) m=12



(g) m=13

In Figure 3(a)-(g) above, at each time step $t = 0...m-1$, the 'score' is calculated as the sum of the number of existing edges which each choice of vertex in the new edge choice hits. In Figure 4(a)-(g), at each time step $t = 0...m-1$, a 'non-greedy' play is represented as a flat line. Thus for all games in 4(a) Bob's moves were considered greedy, and for all games in 4(g) one of Bob's moves was considered non-greedy, occurring at time step $t = 7$.

In Figure 4(g), despite 32 games having the max-size property, just one ordering of greediness choices works to generate the maximum size hypergraph.

In the cases where the depth is minimized, with 7 edges, we can assume Bob has played poorly. Observations on these hypergraphs reveals that Bob puts all his eggs in one basket; he tends to play heavily around a single vertex.

In the max-sized hypergraph with 13 edges, Bob's distribution of edges is more even. This necessitates him playing non-greedily at a key turning point, choosing a vertex which does not hit the maximum number possible remaining vertices. In all the sized 13 hypergraphs, Bob makes a single choice of a non-greedy edge: v_{13} or v_{22}, which participate in just 4 edges, instead of choosing one of the 5-sized edges, v_{31} or v_{41}. He then proceeds to play greedily for the rest of the game. Two example transcripts of max-sized games is shown below in Table 5 and Table 7, and a transcript of all maximum sized games for $r = 4$ is posted on the GitHub repository.

	Alice	Bob	Greedy	Vertex Participation Counts
0	[]	(v1_1, v2_1, v3_1, v4_1)	Greedy	{v1_1: 1, v2_1: 1, v3_1: 1, v4_1: 1}
1	[v1_1, v2_1, v3_1]	(v1_2, v2_2, v3_2, v4_1)	Greedy	{v1_1: 1, v1_2: 1, v2_1: 1, v2_2: 1, v3_1: 1, v3_2: 1, v4_1: 2}
2	[v4_1, v1_1, v1_2]	(v1_3, v2_2, v3_1, v4_2)	Greedy	{v1_1: 1, v1_2: 1, v1_3: 1, v2_1: 1, v2_2: 2, v3_1: 2, v3_2: 1, v4_1: 2, v4_2: 1}
3	[v2_2, v1_1, v1_2]	(v1_3, v2_1, v3_1, v4_1)	Greedy	{v1_1: 1, v1_2: 1, v1_3: 2, v2_1: 2, v2_2: 2, v3_1: 3, v3_2: 1, v4_1: 3, v4_2: 1,
4	[v3_1, v1_2, v1_1]	(v1_3, v2_2, v3_2, v4_1)	Greedy	{v1_1: 1, v1_2: 1, v1_3: 3, v2_1: 2, v2_2: 3, v3_1: 3, v3_2: 2, v4_1: 4, v4_2: 1}
5	[v4_1, v1_3, v1_1]	(v1_2, v2_2, v3_1, v4_2)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 3, v2_1: 2, v2_2: 4, v3_1: 4, v3_2: 2, v4_1: 4, v4_2: 2,
6	[v2_2, v2_1, v1_1]	(v1_3, v2_3, v3_1, v4_1)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 4, v2_1: 2, v2_2: 4, v2_3: 1, v3_1: 5, v3_2: 2, v4_1: 5, v4_2: 2,
7	[v3_1, v2_2, v1_1]	(v1_3, v2_1, v3_2, v4_2)	Not greedy	{v1_1: 1, v1_2: 2, v1_3: 5, v2_1: 3, v2_2: 4, v2_3: 1,

				v3_1: 5, v3_2: 3,
				v4_1: 5, v4_2: 3,
8	[v1_3, v1_2, v1_1]	(v1_4, v2_2, v3_2, v4_1)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 5, v1_4: 1,
				v2_1: 3, v2_2: 5, v2_3: 1,
				v3_1: 5, v3_2: 4,
				v4_1: 6, v4_2: 3}
9	[v4_1, v4_2, v1_1]	(v1_3, v2_2, v3_1, v4_3)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 6, v1_4: 1,
				v2_1: 3, v2_2: 6, v2_3: 1,
				v3_1: 6, v3_2: 4,
				v4_1: 6, v4_2: 3, v4_3: 1}
10	[v1_3, v2_2, v1_1]	(v1_4, v2_1, v3_1, v4_1)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 6, v1_4: 2,
				v2_1: 4, v2_2: 6, v2_3: 1,
				v3_1: 7, v3_2: 4,
				v4_1: 7, v4_2: 3, v4_3: 1}
11	[v3_1, v3_2, v1_1]	(v1_3, v2_2, v3_3, v4_1)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 7, v1_4: 2,
				v2_1: 4, v2_2: 7, v2_3: 1,
				v3_1: 7, v3_2: 4, v3_3: 1,
				v4_1: 8, v4_2: 3, v4_3: 1}
12	[v4_1, v1_3, v1_2]	(v1_1, v2_2, v3_1, v4_2)	Greedy	{v1_1: 2, v1_2: 2, v1_3: 7, v1_4: 2,
				v2_1: 4, v2_2: 8, v2_3: 1,
				v3_1: 8, v3_2: 4, v3_3: 1,
				v4_1: 8, v4_2: 4, v4_3: 1}
13	[v2_2, v2_1, v1_3]	No possible move.		

Table 5: Sample maximally sized hypergraph

	Alice	Bob	Greedy	Vertex Participation Counts
0	[]	(v1_1, v2_1, v3_1, v4_1)	Greedy	{v1_1: 1,
				v2_1: 1,
				v3_1: 1,
				v4_1: 1}
1	[v1_1, v2_1, v3_1]	(v1_2, v2_2, v3_2, v4_1)	Greedy	{v1_1: 1, v1_2: 1,
				v2_1: 1, v2_2: 1,
				v3_1: 1, v3_2: 1,
				v4_1: 2}
2	[v4_1, v1_1, v1_2]	(v1_3, v2_2, v3_1, v4_2)	Greedy	{v1_1: 1, v1_2: 1, v1_3: 1,
				v2_1: 1, v2_2: 2,
				v3_1: 2, v3_2: 1,
				v4_1: 2, v4_2: 1}
3	[v2_2, v1_1, v1_2]	(v1_3, v2_1, v3_1, v4_1)	Greedy	{v1_1: 1, v1_2: 1, v1_3: 2,
				v2_1: 2, v2_2: 2,
				v3_1: 3, v3_2: 1,
				v4_1: 3, v4_2: 1}
4	[v3_1, v1_2, v1_1]	(v1_3, v2_2, v3_2, v4_1)	Greedy	{v1_1: 1, v1_2: 1, v1_3: 3,
				v2_1: 2, v2_2: 3,
				v3_1: 3, v3_2: 2,
				v4_1: 4, v4_2: 1}
5	[v4_1, v1_3, v1_1]	(v1_2, v2_2, v3_1, v4_2)	Greedy	{v1_1: 1, v1_2: 2, v1_3: 3,
				v2_1: 2, v2_2: 4,

				v3_1: 4, v3_2: 2,
				v4_1: 4, v4_2: 2}
6	[v2_2, v2_1, v1_1]	(v1_2, v2_3, v3_1, v4_1)	Greedy	{v1_1: 1, v1_2: 3, v1_3: 3,
				v2_1: 2, v2_2: 4, v2_3: 1,
				v3_1: 5, v3_2: 2,
				v4_1: 5, v4_2: 2}
7	[v3_1, v2_2, v1_1]	(v1_2, v2_1, v3_2, v4_2)	Not greedy	{v1_1: 1, v1_2: 4, v1_3: 3,
				v2_1: 3, v2_2: 4, v2_3: 1,
				v3_1: 5, v3_2: 3,
				v4_1: 5, v4_2: 3}
8	[v3_1, v3_2, v1_1]	(v1_2, v2_2, v3_3, v4_1)	Greedy	{v1_1: 1, v1_2: 5, v1_3: 3,
				v2_1: 3, v2_2: 5, v2_3: 1,
				v3_1: 5, v3_2: 3, v3_3: 1,
				v4_1: 6, v4_2: 3}
9	[v4_1, v4_2, v1_1]	(v1_2, v2_2, v3_1, v4_3)	Greedy	{v1_1: 1, v1_2: 6, v1_3: 3,
				v2_1: 3, v2_2: 6, v2_3: 1,
				v3_1: 6, v3_2: 3, v3_3: 1,
				v4_1: 6, v4_2: 3, v4_3: 1}
10	[v1_2, v1_3, v1_1]	(v1_4, v2_1, v3_1, v4_1)	Greedy	{v1_1: 1, v1_2: 6, v1_3: 3, v1_4: 1,
				v2_1: 4, v2_2: 6, v2_3: 1,
				v3_1: 7, v3_2: 3, v3_3: 1,
				v4_1: 7, v4_2: 3, v4_3: 1}
11	[v3_1, v1_2, v1_3]	(v1_1, v2_2, v3_2, v4_1)	Greedy	{v1_1: 2, v1_2: 6, v1_3: 3, v1_4: 1,
				v2_1: 4, v2_2: 7, v2_3: 1,
				v3_1: 7, v3_2: 4, v3_3: 1,
				v4_1: 8, v4_2: 3, v4_3: 1}
12	[v4_1, v1_2, v1_3]	(v1_4, v2_2, v3_1, v4_2)	Greedy	{v1_1: 2, v1_2: 6, v1_3: 3, v1_4: 2,
				v2_1: 4, v2_2: 8, v2_3: 1,
				v3_1: 8, v3_2: 4, v3_3: 1,
				v4_1: 8, v4_2: 4, v4_3: 1}
13	[v2_2, v2_1, v1_2]	No possible move.		

Table 7: Another sample maximally sized hypergraph

In both example transcripts (and all maximum sized hypergraphs for $r = 4$), the single non-greedy play of the 7th edge enabled Bob to build groupings of edges in parts $v_2 ... v_4$ which eventually reach size 8, 4, and 1 respectively. The vertices which reached size 8 were v_{22}, v_{31},and v_{41}, the vertices which reached size 4 were v_{21}, v_{32}, and v_{42}, and the vertices which only reached size 1 were v_{23}, v_{33}, and v_{43}.

The edges in V_1 follow a slightly different pattern, and differ among the 2 games. However, notice that both spread out among 4 vertices, and the minimum degree is 2. In Game 1, v_{13}= size 7, v_{11}, v_{12}, v_{14} = size 2. In Game 2, v_{12}= size 6, v_{11}, v_{14} = size 2, v_{13} = size 3. This spread, essentially building groups evenly but from the bottom up and spreading remaining vertices out over V_1, is shown to be the optimal strategy for generating a large hypergraph.

We conclude these findings with the suggestion of learning how to generate edges to optimize play. It may be possible to use transcripts of max-depth games in the brute-force $r = 4$ and $r = 5$ case to train a learning algorithm to play optimally. Alternatively, we may come up with some local metric or weighting at each step which can inform the edge choice such that the max-size hypergraph is generated.

4.6 Future algorithmic work

In light of these findings, we conclude that the Alice algorithm is a candidate for efficiently generating a vertex cover with $(r-1)$ elements given any r-partite hypergraph. We have proved it works empirically for $r = 3$ and $r = 4$. We have also proved that using a greedy response algorithm for $r = 3...20$, Alice can always choose a vertex cover for a greedy reactive hypergraph. We now pose the question of whether it is possible to generate an alternate optimal reactive choice of edge which produces the max-size reactive hypergraph to the greedy vertex cover every time, and whether the back-and-forth game would always terminate in this case.

Another open question is that of what a vertex cover algorithm for $|M| \geq 2$ might look like. We pose that it may be possible to take the same approach in choosing vertices; however, generating new edges would be very different, because it would not require that all edges intersect all other edges once.

Additionally, we posit the question of choosing a vertex cover for a hypergraph which is not reactive in nature; that is, any vertices may be chosen at a given time, but the edges must still share the property that the maximum matching is $|M| = 1$. The following section concerns work in this vein.

5 Families of k-Intersecting Hypergraphs and Their Size

We are now concerned about the specific case where all edges intersect all other edges at exactly k points, and specifically, when all edges intersect all edges at exactly 1 point.

Let $k = 1$. Then by the definition of k-intersecting, the intersection of every pair of edges contains exactly one vertex, we have $|e \cap f| = 1,\ \forall\ e, f \in E,\ e \neq f$.

5.1 Properties of an r-Partite 1-intersecting hypergraph with $|M| = 1$

Consider an r-partite hypergraph G which has the additional constraint $|e \cap f| = 1,\ \forall (e, f) \in E$. This additional constraint means that every edge shares one and *exactly* one vertex with every other edge. This hypergraph is said to be *linearly intersecting*.

We want to make a statement about the cardinality of the hypergraph that is possible under these constraints. Notice that, with no bound on the maximum degree of a vertex or the minimum size of a vertex cover, it would be possible to have an infinite number of unique edges: WLOG, choose v_{11} as the vertex which all edges have in common; then the hypergraph is given by the edge set $E = \{[v_{11}, v_{21}..., v_{r1}], [v_{11}, v_{22}, ..., v_{r2}], [v_{11}, v_{23}, ..., v_{r3}], \ ... \ , [v_{11}, v_{2\infty}, \ ... \ , v_{r\infty}]\}$.

Imposing the additional constraint $|C| \geq 2$ bounds the size of the hypergraph to a finite domain. To show this, we first consider a lemma.

Lemma 3.2. *In a 1-intersecting r-partite hypergraph with $|C| \geq 2$, let the r different parts be sets of vertices $V_1...V_r$. Then $\forall\ i = 1...r,\ |V_i| \geq 2$.*

Proof. Suppose not. Then $\exists$ some part i such that v_{i1} is the only vertex in that part. Then all edges must contain v_{i1}. That is, we could select a minimum vertex cover $c* = v_{i1}$, with $|C| = 1$. This contradicts the assumption that $|C| \geq 2$. □

Thus, we may proceed with the additional assumption that all parts have at least two vertices. Notice that the set of hypergraphs that satisfy $|e \cap f| = 1, |C| \geq 2, |M| = 1$ is the empty set for $r = 2$. It is impossible to find a graph which has a vertex cover size of at least 2 in the bipartite graph case when the constraint is imposed that all edges must share an edge with at least one other edge.

This holds with proven theory. By Hall's Theorem, that the size of a minimum vertex cover is equal to the size of maximum matching, $|C| = |M|$, in the bipartite case.

5.2 Bounds on size in a 1-intersecting r-partite hypergraph

When considering what the size of the hypergraph is, one can define several areas of concern:

- The number of edges in the hypergraph
- The number of vertices in the hypergraph
- The maximum degree of any vertex in the hypergraph
- The number of overlaps (that is, the number of vertices where degree is at least 2).

Let the number of edges in the hypergraph be denoted m. The number of vertices in the hypergraph is denoted n.The number of parts is r and the degree of a vertex v is d_v.

Theorem 4. *In a 1-intersecting hypergraph with $|M| = 1$, $|C| \geq 2$, the maximum degree of any vertex is $(r-1)$.*

Proof. Let u be a vertex with maximum degree d_{max}. Then suppose WLOG u is in part V_1. Let D be the set of all edges incident on u; $|D| = d_{max}$. Every edge $e \in D$ intersects every other edge $f \in D$ at exactly one point, and this point of intersection is u; thus $\forall j \neq i$, $|V_j| \geq d_{max}$ and every pair of modified edges $|e - u| \cup |f - u| = \emptyset, (e, f) \in (D)$. By Lemma 3.2 $\exists$ another vertex $u' \in V_i$. $u' \neq u$, and u' nust participate in some edge e'. e' must intersect every edge $d \in D$. There are exactly $(r-1)$ ways to intersect each edge, and exactly $(r-1)$ choices of vertices left to fill in e' (excluding u'), for the parts $V_2...V_r$. We proceed to prove by contradiction. Suppose there are r edges in D. Then WLOG, proceed through each part V_i for each $i = 2...(r)$ in the following way: choose an edge from D and fix its vertex in V_i in the edge e'. No edges in D intersect each other, and only $(r-1)$ vertices may be fixed. Thus the rth edge will not be hit by e', which leads to a contradiction. The maximum number of edges in D is given by $(r-1)$, and d_{max} is therefore bounded above by $(r-1)$. □

With the maximum degree shown to be $(r-1)$, we can now bound the number of possible edges in the hypergraph. This is shown below to be equal to $(r-1)^2$.

Lemma 4.1. *The number of edges m in an r-partite hypergraph is bounded above by $m \leq (r-1)^2$.*

Proof. We use the idea of double counting. Every intersection will be counted twice going over all the vertices, and twice going over all the edges. By the Handshake Lemma, $\sum_v d_v = rm$. Then $\sum_v \binom{d_v}{2} = \binom{m}{2} \implies \sum_v \frac{d_v(d_v-1)}{2} = \frac{m(m-1)}{2}$ follows from the fact that the hypergraph is linear and intersecting (all edges intersect all other edges exactly once). Simplifying,

$$\sum_v d_v(d_v - 1) = m(m-1)$$

$$\sum_v d_v^2 - \sum_v d_v = m(m-1)$$

$$\sum_v d_v^2 = m(m-1) + rm = m(m+r-1)$$

By Lemma 4.1, $\max(d_v) = (r-1)$. Then the absolute bound on these two quantities is

$$\sum_v d_v^2 \leq \sum_v (r-1)^2$$

$$\sum_v d_v \leq \sum_v (r-1)$$

Since each $d_v \leq d_{max}$, it follows that $\sum_v d_v^2 \leq d_{max} \sum_v d_v$, which implies $m(m+r-1) \leq (r-1)rm$. $m+r-1 \leq r(r-1)$, and thus $m \leq (r-1)^2$. □

We now pose the question of the maximum number of vertices possible in such a hypergraph, n. Certainly, the number of vertices is bounded above by $n \leq (r-1)^3$. To see this, simply consider that since the hypergraph is r-uniform, every edge contains r vertices. $m \leq (r-1)^2$, and so it follows that $n \leq r*(r-1)^2$. Since the size of the maximum matching is 1 and the size of the minimum vertex cover is 2, all edges must contain one of the two vertices in the vertex cover; thus every edge has $(r-1)$ free vertices and we can easily state $n \leq (r-1)^3$.

Theorem 5. *If $n \leq r^2$, then Ryser's conjecture holds.*

Proof. Ryser's conjecture states that the size of the minimum vertex cover is at most $(r-1)$ in this case. If $n \leq r^2$, then there exists some part V_i st $|V_i| \leq (r-1)$. Suppose not. Then all V_i would have $\geq r$ vertices. There must be some edge incident on all of the vertices in all of the parts; $d_v \geq 1$. Furthermore, all edges must intersect all other edges. Then for an edge incident on the rth vertex in some part, it must intersect all other edges exactly once. There are at least r edges in the graph in this case; if not, there could not be $\geq r$ vertices in each part. Choose some vertex v_i in V_i. There are at least $r-1$ edges in the graph which do not include that v_i, since there are $r-1$ other vertices in the part. If v_i's incident edge intersects all those edges, then there is some v_j in another V_j which has the same dilemma, and v_i avoids this edge, so we only have $(r-1)$ choices of intersection in V_i for v_j. Repeating for all parts $\neq V_i$, at the rth part, v_r will have no choices of v_i to be incident upon. Therefore we have a contradiction and some part $|V_i| \leq (r-1)$. We choose those $(r-1)$ vertices as our cover. □

We do not know if $n \leq r^2$ in the 1-intersecting graph. In fact we believe it to be false; however, as of writing we are unable to come up with any counterexample where $n \geq r^2$. Thus we conjecture that $n \leq r^2$, which, if shown to be true, proves Ryser. This is left as an open question.

5.3 Tightness on 1-intersecting hypergraph size

From henceforth on, we define the *size* of a hypergraph to be the size of its edge set, m. We have shown in the above that an interesting property of a 1-intersecting r-partite hypergraph is that the number of edges is bounded above by $m \leq (r-1)^2$.

Another avenue of exploration concerns whether or not $m \leq (r-1)^2$ is tight, and under which conditions. To proceed, consider $r=3$, the smallest possible hypergraph with $|M|=1, |C| \geq 2$.

Example. $r=3$, so $(r-1)^2=4$. Consider a hypergraph with six unique vertices, two in each part. The edges $[v_{11}\ v_{21}\ v_{31}]$, $[v_{11}\ v_{22}\ v_{32}]$, $[v_{12}\ v_{21}v_{32}]$, $[v_{12}\ v_{22}\ v_{31}]$ follow the constraints posed. See Figure 5; in the diagram below, each 3-edge is represented by a different shape.

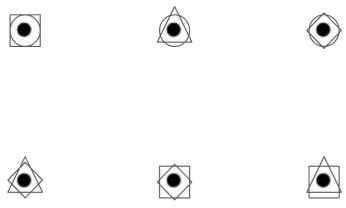


Figure 5: $m=4$ in a maximally sized 1-intersecting hypergraph for $r=3$

Thus, it is possible, in some cases, to achieve tightness on m, chooing $(r-1)^2$ edges for an r-partite hypergraph.

Finding the conditions which generate a hypergraph with $(r-1)^2$ edges poses an interesting question. Additionally, exploring methods of generating a maximally sized hypergraph may provide some light into the algorithmic approach to a general optimal vertex cover. The following sections pose the 1-intersecting maximization problem, explore an analogous problem, prove equivalence, and propose a novel generating algorithm.

5.4 Maximizing size in a 1-intersecting hypergraph

We now consider how we might generate any set of edges larger than $(r-1)$. It is trivial to generate $(r-1)$ edges, all incident on a single vertex v_{11} - a single bundle. All of these edges have the property that they do not overlap at any of the $(r-1)$ remaining vertices. Now we want to introduce a new edge incident on v_{12}. This edge must intersect all of the edges incident in v_{11} exactly once.

Therefore, consider the bundle on v_{11} as an $((r-1)\times(r-1)$ grid. All of the edges (rows) in the bundles on v_{1i} for $i = 2..(r-1)$ must contain exactly one element from each row of the grid. Thus, if the grid is:

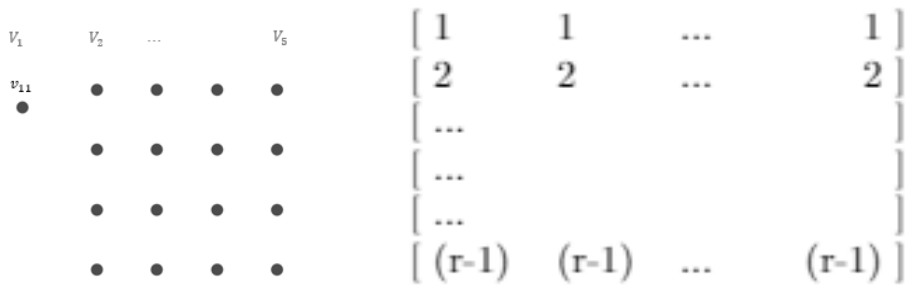


Figure 6: Textual and visual representation of a bundle on v_{11}.

The figure at left represents the construction of the grid, and at right shows the enumeration of vertices in each part $2...(r-1)$. A row in the grid represents part of an edge which also includes v_{11}. Therefore, all edges in the v_{11} bundle contain exactly one intersection. We also know that all edges incident on any other vertex in v_1 must contain All edges incident on $v_{12}...v_{1k}$ must be permutations of $1...(r-1)$.

5.4.1 Definitions

Bundle. Define a *bundle* to be a group of $(r-1)$ edges incident on some vertex $v_{1i} \in V_1$. For all edges in the bundle, the first element is v_{1i}. Then the remaining edges may be arranged in an $(r-1)\times(r-1)$ grid.

For $i = 1$, the bundle is defined as in Table 6 above. For $i = 2...(r-1)$, every row in the grid is a permutation of $1...(r-1)$. We use the shorthand of the vertex which all edges in a bundle are incident on to denote the bundle (v_{1i}).

1-intersecting bundles Two bundles v_{1i} and $v_{1i'}$ are said to be 1-intersecting if $\forall$ rows in v_{1i} indexed by k and $\forall$ row $\in v_{1i'}$ indexed by l, $\exists$ some j st $v_{1i}(k,j) = v_{1i'}(l,j)$.

We would like to find a set of $(r-2)$ bundles such that:

- All rows within a bundle do not share any elements.

- All rows share exactly one element with all rows of other bundles.

The first statement is equivalent to saying all columns in the bundle must also share no elements, and therefore are permutations themselves.

5.5 Analogous problem: mutually orthogonal Latin squares

Mutually orthogonal Latin squares are a well-studied topic. In particular, conjectures about their size have been posed and disproved throughout centuries. Euler famously conjectured that it is impossible to construct 2 Graeco-Latin squares of order 6 and any of the form $n \neq 2 \mod 4$, but his conjecture was disproved by Bose and Shrikhande in 1959 [24] who determined that the only order of Latin square which did not have an orthogonal mate was that of order 2 and 6. Permutations, group theory, and balanced-incomplete-block-designs (BIBDs) have been often used to find maximal sets of mutually orthogonal Latin squares, and have many applications to other fields of combinatorics as well [25].

There are three typical constructions of mutually orthogonal Latin squares, which are well-characterized. The first is the Finite Field construction [12], which is based on an irreducible polynomial root for the finite field GF. We explore this in depth in section 5.5.3. We also make use of a key to the proof of the Generalized Bose Construction [11] to show our reduced-form MOLS are indeed reduced MOLS. First, we pose the problem by providing useful definitions.

5.5.1 Definitions

We first provide some useful notations and definitions.

Latin square. A nth order Latin square is an array of $n \times n$ numbers $1...n$. Specifically, it consists of n sets of the numbers $1...n$ arranged in such a way that no orthogonal (row or column) contains the same number twice.

Reduced Latin square. A reduced Latin square of order n is a Latin square with first row and first column given by $[1, 2, ..., n]$.

Orthogonal Latin squares. A pair of Latin squares is said to be orthogonal if the n^2 pairs formed by juxtaposing the two arrays are all distinct.

Reduced orthogonal Latin squares. We say a set of k different MOLS of order n is reduced if one of the Latin squares is reduced and if the first row in every other Latin square in this set is given by $[1, 2, ..., n]$.

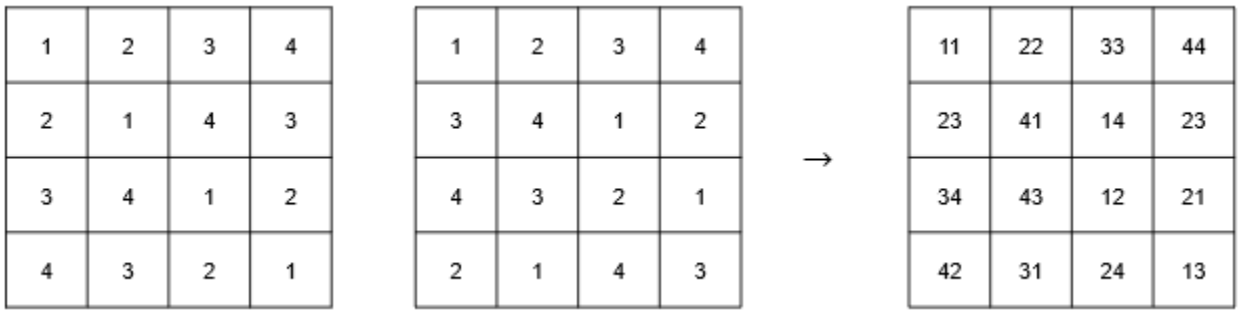


Figure 7: 2 MOLS of order 4, juxtaposed, produce all possible pairs

Remark. A bundle is an example of a Latin square, and vice versa. However, two mutually orthogonal Latin squares are not necessarily two 1-intersecting bundles. To create a bundle v_{1i} from a Latin square A, assign

edges such that each row in A given by A_k is attached to the vertex v_{1i}. Then we could define the bundle: $v_{1i} : [[v_{1i}]\ [A_k]]$.

We now show that the problem of finding $(r-2)$ bundles with the property above is analogous to finding a complete set of mutually orthogonal Latin squares (MOLS).

Theorem 6. *The question of finding $r-2$ 1-intersecting bundles is equivalent to finding $r-2$ mutually orthogonal Latin Squares of order $(r-1)$.*

Corollary 6.1. *If m gives the maximum number of edges in an r-partite 1-intersecting hypergraph, $m = (r-1)^2$ if and only if r is a prime power of the form p^d.*

To guide this chapter, we first state Theorem 6 and then prove it through a series of smaller proofs in Sections 5.5.2 - 5.5.3.

5.5.2 Transforming MOLS to 1-intersecting bundles

We define an operation T on a Latin square M:

$$T : M \to A := \{A_{kj} = i | M_{ij} = k\}$$

Lemma 6.1. *If M is a Latin square, then $A = T(M)$ is a Latin square.*

Proof. If M is a Latin square, fix j. Then every value of i produces a unique k, $M_{ij} = k$. A column in A therefore has the same property that any k produces a unique i. Now fix k. For all pairs (i, j) and (i', j') in which $M_{ij} = k$, $i \neq i' \implies j \neq j'$. A row in A is given by A_k, and therefore every element i must be unique and appear at index j. Thus every row in A is a permutation, and every column in A is a permutation; no row or column contains two of the same elements. Therefore A is a Latin square. □

Theorem 7. *If M and N are two reduced mutually orthogonal Latin squares, then $A = T(M)$ and $B = T(N)$ are two 1-intersecting bundles with 1's on their diagonals.*

Proof. Let M and N be two reduced Latin squares of order $(r-1)$. Let an entry in M be $M_{ij} = k$. Let an entry in N be $N_{i'j} = l$. $l = j$ if $i \neq i'$ by the definition of mutually orthogonal. Two reduced Latin squares have unique row and column elements for all rows and columns except the first; that is, $M_{1j} = N_{1j} = j$; $M_{ij} \neq N_{ij} \neq j, \forall\ i = 2...n, j = 1...n$. Consider an arbitrary row in A indexed by k and an arbitrary row in B indexed by l. NTS exactly one element is shared between the two; that is, $\exists$ some j st $A_{kj} = B_{lj}$ for any two $(k, l) \in (r-1)^2$. By Lemma 6.1 above, A and B are Latin squares. Since $k \in \{1...(r-1)\}, l \in \{1...(r-1)\}$, k appears somewhere in every column of M and l appears somewhere in every column of N. Since M and N are mutually orthogonal, $\exists$ only one choice of indices (a, b) st $M_{ab} = k$ and $N_{ab} = l$. We then have $A_{kb} = a$ and $B_{lb} = a$. Thus $\exists$ exactly one choice of $j = b$ st $A_{kj} = B_{lj}$. □

It follows from here that the exercise of maximizing the number of orthogonal Latin squares is equivalent to maximizing the number of playable 1-intersecting bundles.

Theorem 8. *For all A_i generated from reduced form MOLS, the intersecting element between two bundles A_i and A_i' in a row r is given by the rth element in the row, which is 1: $A_i(r, r) = 1 = A_{i'}(r, r)$.*

Proof. If M is in reduced form, then $M_{1j} = j \forall j = 1...(r-1)$. This transforms to $T(M_i) = A_i \implies A_i(j, j) = 1$. Since all M_i are reduced form, all $A_i(j, j) = 1$. All A_i will be Latin squares with 1's on the diagonal. Therefore 1 must be the intersecting element between the element in row j with the element in row j of some other $A_{i'}$. □

5.5.3 Maximal set of mutually orthogonal latin squares

Definition. A maximal set of mutually orthogonal Latin squares (MOLS) is a set of k n-order MOLS such that it is impossible to extend the set to a set of $(k+1)$ MOLS of order n.

The maximal number of mutually orthogonal Latin squares of an order $(r-1)$ is given by

$$N(r-1) = \max(k : \exists(k) MOLS(r-1))$$

Hicks, Mullen, Storme, and Vanpoucke (2018) explore extensively the number of mutually orthogonal Latin squares for different factorizations of primes in [8]. This begins with the simple proof that $N(n) \leq (n-1)$, and then delves into complicated constructions to show $N(r-1) = r-2$ if and only if $r-1$ is a prime or a prime power. Interestingly, all proofs we find of this relies on different constructions of MOLS to show tightness in conjunction with an upper bound, rather than a numerical approach. This is one additional way our construction (posed in Section 5.6 may be of use.

Lemma 8.1. *For every $r \geq 2$, $N(r-1) \leq (r-2)$. [10]*

Proof. WLOG consider 2 MOLS L and M in reduced form. We can place any MOLS into reduced form without affecting the orthogonality of them by performing a permutation on the alphabet by [10]. We then consider the element $L(2,1) = a$ and $M(2,1) = b$. Because the element $L(1,1) = 1$ and $M(1,1) = 1$, and $a \neq b$ by the definition of orthogonal, we have $a \neq b \neq 1$. This means that a may only take on $n-1$ values, and since a group of MOLS cannot have the same symbol at the position $(2,1)$ (WLOG), the number of MOLS is bounded above by $(n-1)$. □

Theorem 9. *If $q = p^d$ is a prime power, then $N(q) = (q-1)$. [8]*

Proof. Any prime power $q = p^d$ may be expressed as a finite field of elements $GF(p^n)$. Any prime may be represented as $GF(p) = 0, 1, 2, ..., p-1$. This is equivalent to our choice of a above. $GF(p^n)$ is the field of equivalence classes of polynomials whose coefficients belong to $GF(p)$; that is, the elements of $GF(p^n)$ can be written as a polynomial using elements only in $GF(p)$. $GF(q) = 0, 1, 2, f(x), ...q-1$. We can take the modulus of any of the polynomials for a given row index, $f(x) = x_d p^{d-1} + x_{d-1} p^{d-2} + ...x_1 p + x_0$. Then construct the Latin squares as follows: $M_a(x,y) = ax + y \mod f(x)$. The function $f(x)$ maps all values of x, the row index, to a unique composition of the prime raised to powers lower than d. We can follow a similar reasoning as in Lemma 9.1 to conclude these $(q-1)$ Latin squares are mutually orthogonal. [8] □

We next show that tightness is possible in the case where the order $r-1$ of the Latin square of interest is a prime, $r-1 = p$ using the modulo operator, and in fact that it is possible when $r-1$ is a prime power using a finite field construction and a primitive polynomial function such that every element at every (i,j) coordinate in every square differs from all others aside from the first row.

Lemma 9.1. *If the Latin square order of interest $r-1$ is some prime $r-1 = p$, then it is possible to construct a set of $(p-1)$ MOLS. [10]*

Proof. Consider the case where p is a prime number $(d = 1)$. Then let $a = 0...p-1$. We can construct exactly $p-1$ mutually orthogonal Latin squares as follows, using the fact that p is a prime. Let $M_a(x,y) = ax + y \mod p$. Then consider two arbitrary elements in the same column y indexed by x and x'. $M_a(x,y) = M_a(x',y) \iff ax + y = ax' + y$. Since $a \in [1, p-1]$, $ax + y = ax' + y$ generates a unique value for every x. Then $ax + y = ax' + y \iff x = x'$; that every column y has every element different from every other. Similarly, consider two elements in the same row x. $M_a(x,y) = M_a(x,y') \iff ax + y = ax' + y \iff y = y$ by similar logic; all elements in the same row differ. Now for two squares M_a and M_b, NTS every choice of indices (x,y) generates a different pairing. Suppose the same ordered pair appears in slots (x,y) and (x',y') after superimposition. Then $ax + y = ax' + y'$ and $bx + y = bx' + y'$. Subtracting one equation from the other, $ax - bx = ax' - bx' \implies (a-b)x = (a-b)x'$. As $a \neq b$, this holds only when $x = x'$ and $y = y'$. □

Remark. As an addendum, all MOLS generated in this way will have the property $M_{1y} = y$; the first row will be $[12....p-1]$ and therefore all MOLS constructed in this way will be reduced Latin squares.

We apply the approach in Lemma 9.1 to the case where the order of the Latin square of interest is a prime power, $q = p^d$, and use the finite field construction to prove tightness in this case. Note that $GF(q)$ is cyclic, and there exists an element α, such that the $q-1$ non-zero elements of $GF(q)$ are 1, $\alpha, \alpha^2, ..., \alpha^{q-2}$. Then $\alpha^{q-1} = 1$. This means we can always find a valid $(q-1)$ cycle for any q prime power. To illustrate this construction, we show an example construction for $q = 16$ below.

Example. Construction of 15 MOLS(16).
In this case, $p = 2, d = 5$. We construct: $GF(p) = \{0, 1\}$ and thus all elements in $GF(16)$ may be expressed as $x_0 + x_1\alpha + x_2\alpha^2 + x_3\alpha^3$,where $\alpha^4 = \alpha + 1$ is an irreducible polynomial.

It follows that the operations + and $\times$ in $GF(16)$ are:

$$(a + b\alpha + c\alpha^2 + d\alpha^3) + (e + f\alpha + g\alpha^2 + h\alpha^3) = (a + e) + (b + f)\alpha + (c + g)\alpha^2 + (d + h)\alpha^3$$

$$\begin{aligned}(a + b\alpha + c\alpha^2 + d\alpha^3)(e + f\alpha + g\alpha^2 + h\alpha^3) =& (ae + bh + cg + df) \\ &+ (af + be + bh + cg + df + ch + dg)\alpha \\ &+ (ag + bf + ce + ch + dg + dh)\alpha^2 \\ &+ (ah + bg + cf + de + dh)\alpha^3.\end{aligned}$$

[26]
Next, define $M_a(x, y) = ax + y \mod f(x)$ for every $a = 0, ..., q-1$.

We define the values of a as per their binary codes in polynomial form (since we are basing on $GF(2)$:

Decimal value	Binary value	Polynomial
0	0000	0
1	0001	1
2	0010	x
3	0011	x + 1
4	0100	x^2
5	0101	$x^2 + 1$
6	0110	$x^2 + x$
7	0111	$x^2 + x + 1$
8	1000	x^3
9	1001	$x^3 + 1$
10	1010	$x^3 + x$
11	1011	$x^3 + x + 1$
12	1100	$x^3 + x^2$
13	1101	$x^3 + x^2 + 1$
14	1110	$x^3 + x^2 + x$
15	1111	$x^3 + x^2 + x + 1$

Table 8: $GF(16)$ in polynomial form

We use these polynomials along with the defined operations to generate multiplication tables for every 2 polynomial representations of $0...q-1$. These are shown below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	4	6	8	10	12	14	3	1	7	5	11	9	15	13
0	3	6	5	12	15	10	9	11	8	13	14	7	4	1	2
0	4	8	12	3	7	11	15	6	2	14	10	5	1	13	9
0	5	10	15	7	2	13	8	14	11	4	1	9	12	3	6
0	6	12	10	11	13	7	1	5	3	9	15	14	8	2	4
0	7	14	9	15	8	1	6	13	10	3	4	2	5	12	11
0	8	3	11	6	14	5	13	12	4	15	7	10	2	9	1
0	9	1	8	2	11	3	10	4	13	5	12	6	15	7	14
0	10	7	13	14	4	9	3	15	5	8	2	1	11	6	12
0	11	5	14	10	1	15	4	7	12	2	9	13	6	8	3
0	12	11	7	5	9	14	2	10	6	1	13	15	3	4	8
0	13	9	4	1	12	8	5	2	15	11	6	3	14	10	7
0	14	15	1	13	3	2	12	9	7	6	8	4	10	11	5
0	15	13	2	9	6	4	11	1	14	12	3	8	7	5	10

Figure 8: Addition and Multiplication under $GF(16)$

Finally, we use the construction to generate the 15 mutually orthogonal Latin squares and corresponding 1-intersecting bundles.

$$M_a(x, y) = ax + y \mod f(x)$$

Where $f(x)$ is a primitive polynomial. The root of this polynomial, α, is a root of the finite field $GF(q)$. Any primitive polynomial of order d may be used as the modulo function $f(x)$. In the example below, the primitive polynomial used was $x^4 + x + 1$.

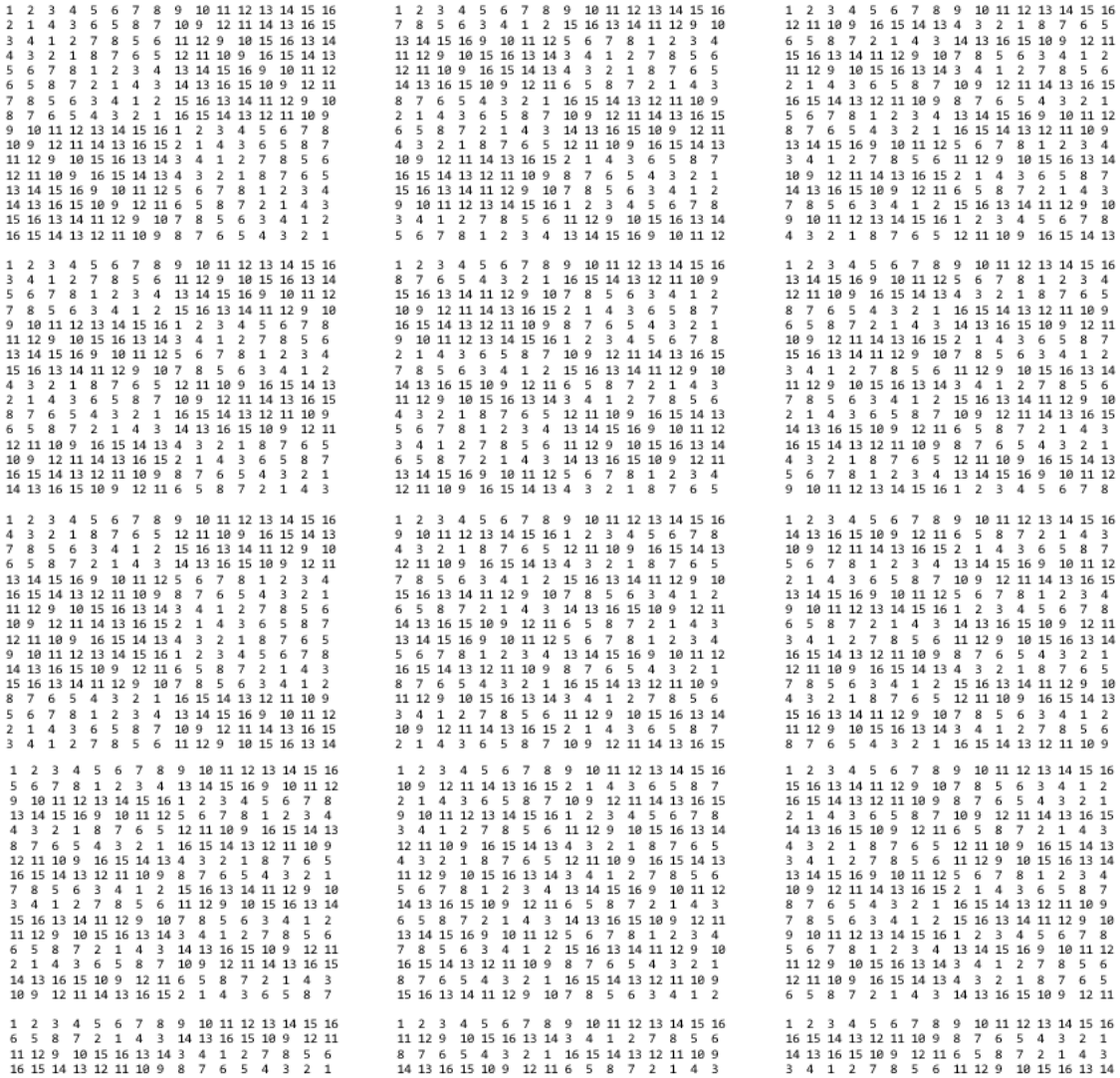


Figure 9: 15 mutually orthogonal Latin squares of order 16

We apply the transformation T to generate the final 1-intersecting bundles.

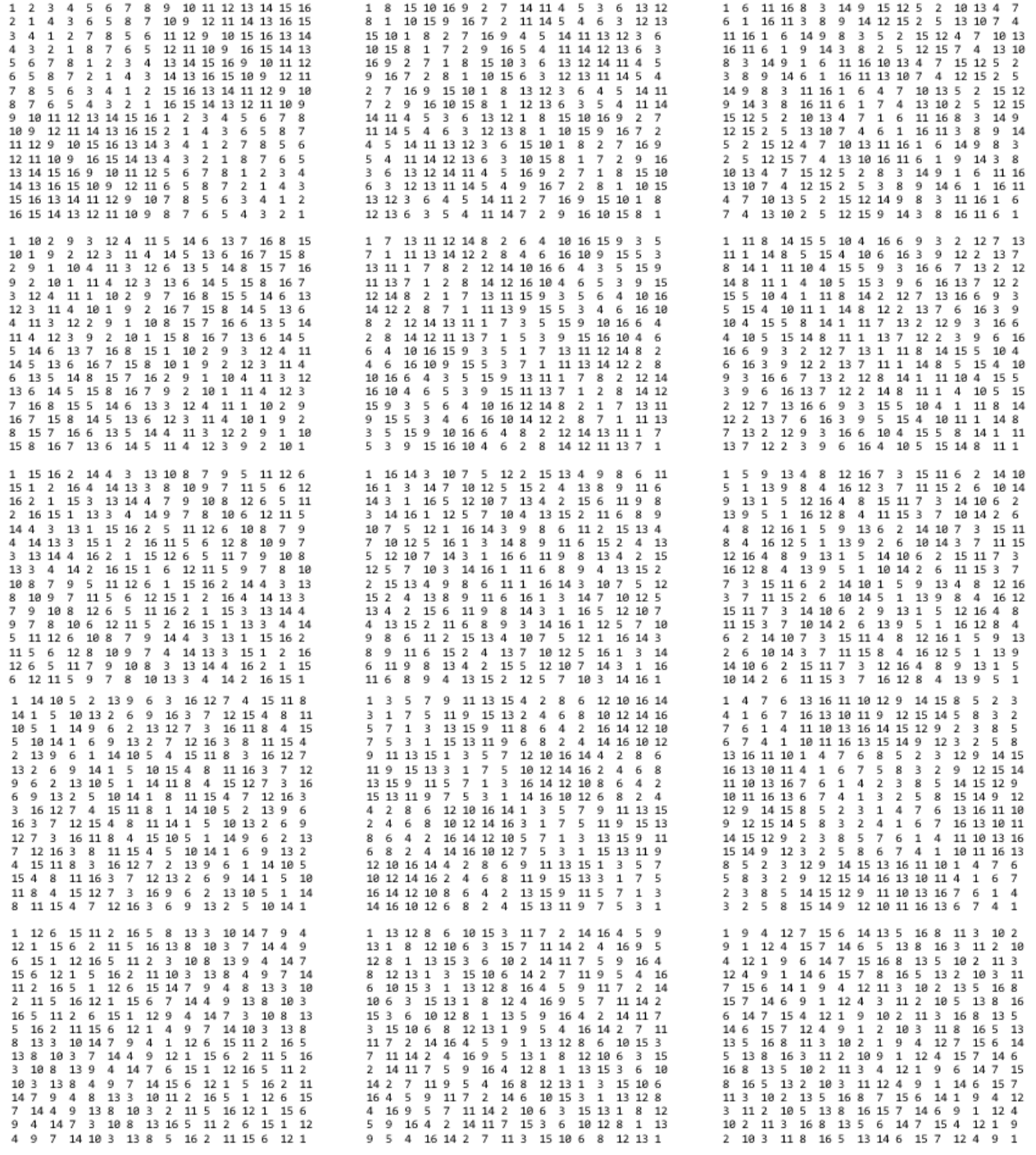


Figure 10: 15 1-intersecting bundles of order 16

Note that all the 1-intersecting bundles have the property that 1's occur along their diagonal and in fact for the primitive polynomial seed chosen, all 1-intersecting bundles are perfectly symmetric.

5.6 An alternative approach to generating $(r-2)$ 1-intersecting bundles

For an edge in a bundle, we number the vertices according to the integers $1...(r-1)$. This is equivalent to $Z_{r-1} = \{1, 2, ..., (r-1)\}$, so each edge may be thought of as a permutation.

Permutation. A permutation of size n is a one-to-one mapping of elements in Z_n to themselves. We denote such a permutation as $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$.

Permutation cycle. A *permutation cycle* or *orbit* of a permutation π_i is some subset of elements in π such that elements map onto each other. Let an arbitrary cycle of size k be denoted $(a_1, a_2, ...a_k)$ such that $\pi_i(a_j) = a_{j+1}$ for all $j = 1...(k-1)$ and $\pi_i(a_k) = (a_1)$.

A permutation with a single fixed element may arbitrarily assign that element to the first index, 1. Then 1 will be in the first position and it is possible to generate a maximum of $(r-2)$ permutations such that no permutation intersects any other permutation at any index besides the first. Permutations must take advantage of cyclic breakdown to achieve this; that is, every element must necessarily be in every position exactly once.

Consider the construction below. We wish to generate $(r-2)$ 1-intersecting bundles such that:

- No two rows in the same bundle share any elements.
- Every row in any bundle intersects every row in every other bundle exactly once.

WLOG, fix the intersection of the first row of every bundle with every other bundle to be the number 1 in the first position. We now define two objects: σ, the transformation from the first row of one bundle to the first row of the next, and Π, the set of permutations within a bundle transforming its first element to each subsequent row.

Π							
1	2	3	4	5	6	7	8
5	6	7	8	1	2	3	4
7	8	5	6	3	4	1	2
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
8	7	6	5	4	3	2	1
6	5	8	7	2	1	4	3
2	1	4	3	6	5	8	7

σ							
1	5	7	3	4	8	6	2

Figure 11: Example Π and σ for $r - 1 = 8$

Let the transformation within a bundle be given by Π and each row defined by π_i, i.e., row $1 = \pi_1 = I$, row $2 = \pi_2$, ... row $(r-1) = \pi_{r-1}$. Let the permutation which fixes the first element and assigns every element to every other element be denoted σ. Then we have the relationship that $\sigma = \Pi(1)$, the ordering of the first elements of permutations $\pi_1(1)...\pi_8(1)$ in a given bundle.

Then we want the permutations to have the property: For every i-indexed row in M, i'-indexed row in M', $M_{ij} = M'_{i'j}$ for exactly one j.

We construct and verify the $(r-2)$ bundles as follows.

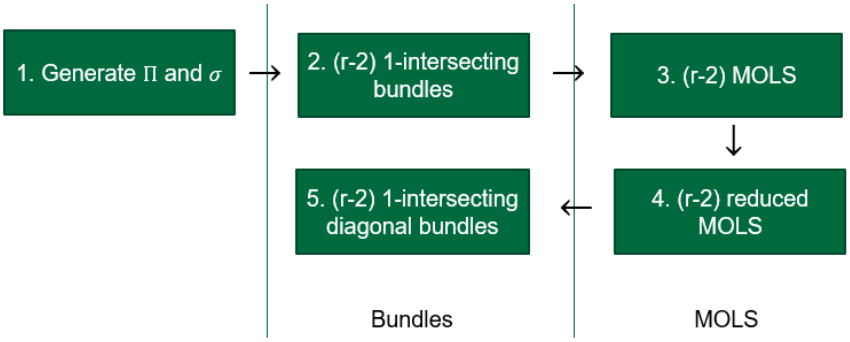


Figure 12: The process of generating and verifying (r-2) 1-intersecting bundles

We outline the process of each step in sections 5.6.1 - 5.6.4 below.

5.6.1 Novel construction of permutation seed for 1-intersecting bundle

Let C_p be some cyclic permutation on the symbols $1..p$ and the set of permutations $\{I, C_p, C_p^{-1}\}$ is strictly transitive. For prime powers $q = p^d$ of the form such that $pd+1$ is a prime power, construct σ and Π in the following way.

Transitive permutation set. Given a set of permutations $1, 2, ..., k$ on a set S, we say that the set of permutations $\{\pi_1, \pi_2, ..., \pi_i, ..., \pi_k\}$ is transitive on S if for every ordered pair of elements $(a, b) \in S$, there exists at least one π_i for which $\pi_i(a) = b$.

Sharply transitive. A permutation set for which there is exactly one π_i which maps a to b is called *sharply transitive*.

We generate the permutations $\Pi = \pi_i$ and σ according to the following algorithm:

- Generate C_p, a cyclic permutation on the elements of $1...(p)$ given in standard form: $[(p)\ 1\ 2\ ...\ (p-1)]$..
- Generate a sequence which has elements given by the powers of p from p^0 to $p^{(d-1)}$. Let the powers of p be a set $p*$ in increasing order, $p* = \{1, p, p^2, ...\}$

  Create the sequence seq of size $(r-1)$.

  For $i = 0...(r-1)$:

  For each $pow \in p*$, if $(i \mod pow == 0)$ set $seq[i] = pow$.
- Let the first row of Π be given by I. For each row subsequent:

  Let the divisor for a permutation π_i be given by $seq[i]$.

  Traverse the previous permutation π_{i-1} and apply a local permutation at the level given by $seq[i]$.
- Set the permutation $\sigma = \pi_i(1)$.

We are able to apply this to powers of 2 or 3, since these powers have strictly transitive cycles $\{2, 1\}$ and $\{3, 1, 2\}$. However, we limit the problem to powers of 2 for simplicity.

Example. $r = 8$. Π is generated in the following way for the $(r-1) = 8$ case. Then $p = 2$, $d = 3$.

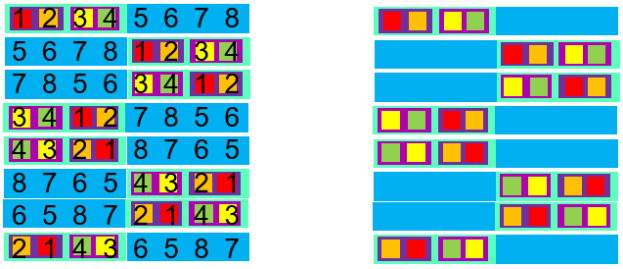


Figure 13: Example of construction of Π for $r-1=8$

Colors above indicate the local swapping of elements. The resulting permutation gives both the ordering of Π and the value of a transformation σ. σ is given by the first column,

$$\sigma = [\ 1\ 5\ 7\ 3\ 4\ 8\ 6\ 2\ ]$$

In the $p=2$ case, the resulting σ is the in-order traversal of a binary tree for $p=2$. To see this, create a tree with d levels ($d=3$ in the $(r-1)=8=2^3$ case). At each node, assign the value 2^{l-1}, with $l=1$ at the root. On each edge, assign an operator $+$ to each leftward edge and $-$ to each rightward edge. Then, perform an in-order traversal of the tree.

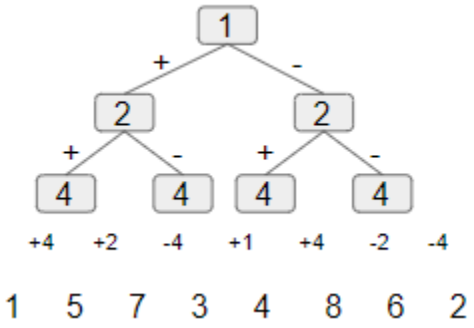


Figure 14: Example permutation construction for $r=8$

This transformation σ has the property that $\Sigma = \{\sigma, \sigma^2, \sigma^3, \sigma^4, \sigma^5, \sigma^6, I\}$ has one fixed element, $\sigma_1 = 1$, and otherwise the elements are sharply transitive with a 7-cycle. Generally, under this construction, $\{\sigma, \sigma^2, ...\sigma^{r-3}, I\}$ has one fixed element and one $(r-2)$ sized orbital. That is, for every $(a,b) \in (2...r-1)^2$, $\exists$ a unique σ_i for which $\sigma_i(a) = b$.

We create the $(r-2)$ 1-intersecting bundles applying Π to each of the elements of $\Sigma = \{\sigma, \sigma^2, \sigma^3, \sigma^4, \sigma^5, \sigma^6, I\}$. Any power of sigma can be expressed as the sum of any other powers $\mod (r-2)$. For example, $\sigma^6(\sigma) = \sigma^7 = \sigma^0 = I$. Similarly, $\sigma^3(\sigma^4)) = I$, $\sigma^3(\sigma^2) = \sigma^5$, and $\sigma^3(\sigma^5) = \sigma^8 = \sigma^1$. The resulting squares are $\{\Pi(I), \Pi(\sigma), \Pi(\sigma^2), \Pi(\sigma^3), \Pi(\sigma^4), \Pi(\sigma^5), \Pi(\sigma^6)\}$.

We continue the sample illustration below.

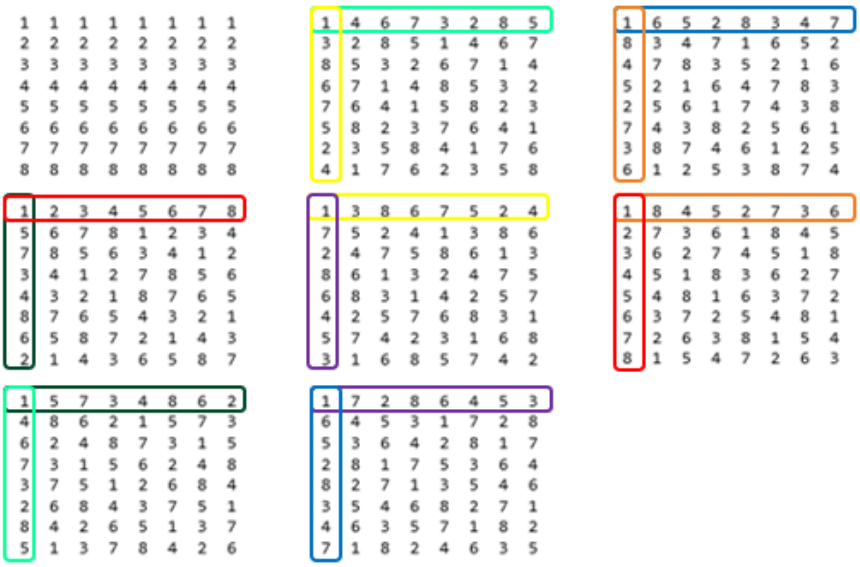


Figure 15: Generating (r-2) 1-intersecting bundles

We pose in Conjecture 22 that our construction yields the following property: *For any* $\pi_i \in \Pi$, $\pi_i(j) = k \iff \pi_i(k) = j$. *Thus* $\pi_i^{-1} = \pi_i$ *and* $(\pi_i)(\pi_i) = I$, *the identity.*

Assuming this conjecture to be true, we prove the following corollaries about applying Π, Corollary 9.1 and 9.2.

Corollary 9.1. *For any permutation* p, *if applying* Π *to* p *forms a matrix with each row given by* $\pi_i p$, *then* $\pi_i p(j) = k \iff \pi_i p^{-1}(k) = p(j)$.

Proof. Let $\pi_i p(j) = k$. By the definition of an inverse, if $\tau[j] = k$, then $\tau^{-1}(k) = j$. Consider $\tau = \pi_i p$. The inverse of the permutation composition $\pi_i p$. Then $(\pi_i p)^{-1} = (p^{-1}\pi)$; to see this, $\pi_i p p^{-1}\pi_i = \pi_i \pi_i = I$ by Conjecture 22. We then have that if $\pi_i p(j) = k$, $p^{-1}\pi_i(k) = j$. Then consider $\pi_i(a) = b$ for some a, b. $(\pi_i p)(a) = c$, and $(\pi_i p)(b) = d$. Then have that $p(b) = c$ and $p(a) = d$, $p^{-1}(d) = a, p^{-1}(c) = b$. Let $a = p(j)$; $k = c$. Then $p^{-1}(k) = b$. $\pi_i(b) = a = p(j)$. □

Corollary 9.2. *For any row in any* A, *which is the result of applying* Π *to some power of* σ *given as* σ^x, $\pi_i(\sigma^x)(\sigma(i)) = 1$.

Proof. Let σ^x be a permutation p. There exists a bijective mapping $f : p \to I$ by the definition of a permutation. Thus taking the elements of p to be the natural elements of I, by Conjecture 22, we have $\pi_i(j) = p(j)$ iff $p(j) = j$ (or in other words, if $p = I$). By the construction, σ^x has the first element fixed as $\sigma^x(1) = 1$ and all other elements $j = 2..n$ are guaranteed $\sigma^x(j) \neq j$ if $\sigma^x \neq I$, by property of the orbital. So $p(j) = j$ is only true for $\sigma^x(1) = 1$. Thus for a row $\pi_i\sigma^x$, $(\pi_i\sigma^x)(j) = \pi_i(j)$ when $j = \sigma(i)$ and $\pi_i(j) = 1$. □

Corollary 9.3. *For any* i*th row in any* $A = \Pi\sigma^x$, *which is the result of applying* Π *to some power of* σ *given as* σ^x, *and any other row* i' *in some* $A' = \Pi\sigma^y$, *which is which is the result of applying* Π *to some power* σ^y, *let* $\sigma^y = \sigma^{x+N}$. *Then two rows at the same row index* i *intersect only at their element* k *at index* j *given by* $j = \sigma(i)$: $(\pi_i\sigma^x)(\sigma(i)) = k = 1 = (\pi_i\sigma^y)(\sigma(i))$.

Proof. Let $\sigma^y = \sigma^{x+N}$. Then the transformation $\sigma^x \rightsquigarrow \sigma^y$ is defined as $p = \sigma^N$. $\pi_i\sigma^x = \pi_{i'}\sigma^y$ if $\pi_i\sigma^N = \pi_i$; we can project the transformation back to take $\sigma^x = I$ WLOG. Then take $p = \sigma^N$. We have that $\pi_{i'}\sigma^N = \pi_i$ in two cases. If $i = i'$, by Corollary 9.1, $\pi_i p(j) = k \iff \pi_i p^{-1}(k) = p(j)$. Since $\sigma^N = p$, we know $p^{-1}(k) = p(j)$ iff $\sigma^N(i) = \sigma^{r-1-N}(i)$. These are inversely related and since $r - 1$ must be odd since r is a

power of 2, σ^N and σ^{r-1-N} are two powers $\in \Sigma$ that are not the same. Thus they share only the element 1 by the construction, and the statement holds only when $\sigma(i) = 1$. □

Corollary 9.4. *For any ith row in any $A = \Pi\sigma^x$, which is the result of applying Π to some power of σ given as σ^x, and any other row i' in some $A' = \Pi\sigma^y$, which is which is the result of applying Π to some power σ^y, let $\sigma^y = \sigma^{x+N}$. Then $(\pi_i\sigma^x)(\sigma(i)) = (\pi_{i'}\sigma^y)(\sigma(i'))$ iff $i = \sigma^N(i')$.*

Proof. Again, take $\sigma^y = \sigma^{x+N}$. Then the transformation $\sigma^x \rightsquigarrow \sigma^y$ is defined as $p = \sigma^N$. As in 9.3, we know $\pi_i\sigma^x = \pi_{i'}\sigma^y$ if $\pi_i\sigma^N = \pi_i$ and take $p = \sigma^N$. We now consider the case where $i \neq i'$.

□

To summarize the above:

- Setting $\sigma = \pi_i(1)$ means $\forall\pi_i$, $\pi_i(\sigma(i)) = 1$.
- Additionally, for all elements in any bundle $A = \Pi(\sigma^x(i))$, $\pi_i(\sigma(i)) = 1$ and $\pi_i(j \neq \sigma(i)) = (\pi_i\sigma^x)(j)$.
- For $x \neq y$, $\sigma^x(j) = \sigma^y(j) \iff j = 1$ and $(\pi_i\sigma^x)(j) = (\pi_i\sigma^y)(j) \iff \sigma(i) = 1$. The ith row of any A intersects with the ith row of any other A' iff $A(i,j) = 1 = A'(i,j)$, so any two rows in any two bundles intersect only at the element 1. Thus all rows given by the same index overlap.
- For all rows i and i' in two different bundles, $i \neq i'$, the bundles overlap at some unique k given by the transformation between the two bundles, σ^N, applied to

Let the mapping from σ_i to σ_j be f_{ij}. Then, since σ_i, σ_j are both powers of σ, the mapping is also some power of σ and is also given by some inverse power of σ.

5.6.2 Transforming 1-intersecting bundles to mutually orthogonal Latin squares

We now show that for the bundle construction above, the Latin squares produced in transforming the bundles to squares will be mutually orthogonal.

Recall that the transform

$$T : M \to A := \{A_{kj} = i | M_{ij} = k\}$$

Is a bijective mapping from a Latin square M to a 1-intersecting bundle A which is also a Latin square. Then we can take

$$T^{-1} : A \to M := \{M_{kj} = i | A_{ij} = k\}$$

Applying this on the bundles $\Pi(\Sigma)$, we have the following. In the lemmas and theorems 10 - 11.1, we refer to the set of MOLS as $M = \{M^1, M^2, ...M^{(r-2)}\}$ and denote $M^1 = T^{-1}\Pi$, $M2 = T^{-1}\sigma\Pi$, and so forth. We show:

- **Theorem 10.** The first row of all M is the permutation σ^{-1}.
- **Theorem 11.** M^1, the transform $T^{-1}\Pi$, is symmetric.
- **Corollary 11.1.** M^1's first column is σ^{-1}.
- **Theorem 12.** M^1 has 1's along its diagonal.

Theorem 10. *The first row of all M is the permutation σ^{-1}. $M_{1j} = \sigma^{-1}(j)$ for all j, M.*

Proof. Recall that each A may be represented $\Pi(\sigma^x)$. For each row $\pi_i\sigma^x$ in any A which is indexed by i, by 9.2, $\pi_i\sigma^x(\sigma(i)) = 1$. Then we can write $A_{i\sigma(i)} = 1$. In M, $M_{1\sigma(i)} = i$. Therefore, $M_{1j} = \sigma^{-1}(j)$ for all j. □

Theorem 11. *The first MOLS, M^1, is symmetric by the construction $M^1 = T^{-1}(\Pi)$; $M^1_{ij} = M^1_{ji}\ \forall\ (i,j)$ coordinates.*

Proof. For all $\pi_i \in \Pi$, $\pi_i(a) = b \iff \pi_i(b) = a$. Let $\pi_i(j) = A_{ij} = k$. Then $A_{ik} = j$. This means $M_{kj} = i$ and $M_{jk} = i$. Thus M^1 is symmetric. □

Corollary 11.1. *In M^1, $M^1_{i1} = \sigma^{-1}(i)$ for all i.*

Proof. This follows from Theorem 11. $\Pi_{ij} = 1$ when $j = \sigma(i)$. Thus $M^1_1 j = i$ when $j = \sigma(i)$, and when $j = 1$, $\sigma(i) = 1 \implies \sigma^{-1}(j) = i$. □

Theorem 12. *M^1 has 1's along its diagonal, $M_{ii} = 1$.*

Proof. By T^{-1}, any element $A_1 j = k$ will map to $M_k j = 1$. This means that for $A_1 j = I$, $M_{jj} = 1$. □

We continue for our example, $(r-2) = 8$, taking the elements of $\Pi\Sigma$ to M:

```
Bundle:                 Corresponding MOLS:     Bundle:                 Corresponding MOLS:
1 2 3 4 5 6 7 8         1 8 4 5 2 7 3 6         1 7 2 8 6 4 5 3         1 8 4 5 2 7 3 6
5 6 7 8 1 2 3 4         8 1 5 4 7 2 6 3         6 4 5 3 1 7 2 8         4 5 1 8 3 6 2 7
7 8 5 6 3 4 1 2         4 5 1 8 3 6 2 7         5 3 6 4 2 8 1 7         6 3 7 2 5 4 8 1
3 4 1 2 7 8 5 6         5 4 8 1 6 3 7 2         2 8 1 7 5 3 6 4         7 2 6 3 8 1 5 4
4 3 2 1 8 7 6 5         2 7 3 6 1 8 4 5         8 2 7 1 3 5 4 6         3 6 2 7 4 5 1 8
8 7 6 5 4 3 2 1         7 2 6 3 8 1 5 4         3 5 4 6 8 2 7 1         2 7 3 6 1 8 4 5
6 5 8 7 2 1 4 3         3 6 2 7 4 5 1 8         4 6 3 5 7 1 8 2         8 1 5 4 7 2 6 3
2 1 4 3 6 5 8 7         6 3 7 2 5 4 8 1         7 1 8 2 4 6 3 5         5 4 8 1 6 3 7 2

Bundle:                 Corresponding MOLS:     Bundle:                 Corresponding MOLS:
1 5 7 3 4 8 6 2         1 8 4 5 2 7 3 6         1 6 5 2 8 3 4 7         1 8 4 5 2 7 3 6
4 8 6 2 1 5 7 3         6 3 7 2 5 4 8 1         8 3 4 7 1 6 5 2         5 4 8 1 6 3 7 2
6 2 4 8 7 3 1 5         5 4 8 1 6 3 7 2         4 7 8 3 5 2 1 6         7 2 6 3 8 1 5 4
7 3 1 5 6 2 4 8         2 7 3 6 1 8 4 5         5 2 1 6 4 7 8 3         3 6 2 7 4 5 1 8
3 7 5 1 2 6 8 4         8 1 5 4 7 2 6 3         2 5 6 1 7 4 3 8         4 5 1 8 3 6 2 7
2 6 8 4 3 7 5 1         3 6 2 7 4 5 1 8         7 4 3 8 2 5 6 1         8 1 5 4 7 2 6 3
8 4 2 6 5 1 3 7         4 5 1 8 3 6 2 7         3 8 7 4 6 1 2 5         6 3 7 2 5 4 8 1
5 1 3 7 8 4 2 6         7 2 6 3 8 1 5 4         6 1 2 5 3 8 7 4         2 7 3 6 1 8 4 5

Bundle:                 Corresponding MOLS:     Bundle:                 Corresponding MOLS:
1 4 6 7 3 2 8 5         1 8 4 5 2 7 3 6         1 8 4 5 2 7 3 6         1 8 4 5 2 7 3 6
3 2 8 5 1 4 6 7         7 2 6 3 8 1 5 4         2 7 3 6 1 8 4 5         2 7 3 6 1 8 4 5
8 5 3 2 6 7 1 4         2 7 3 6 1 8 4 5         3 6 2 7 4 5 1 8         3 6 2 7 4 5 1 8
6 7 1 4 8 5 3 2         8 1 5 4 7 2 6 3         4 5 1 8 3 6 2 7         4 5 1 8 3 6 2 7
7 6 4 1 5 8 2 3         6 3 7 2 5 4 8 1         5 4 8 1 6 3 7 2         5 4 8 1 6 3 7 2
5 8 2 3 7 6 4 1         4 5 1 8 3 6 2 7         6 3 7 2 5 4 8 1         6 3 7 2 5 4 8 1
2 3 5 8 4 1 7 6         5 4 8 1 6 3 7 2         7 2 6 3 8 1 5 4         7 2 6 3 8 1 5 4
4 1 7 6 2 3 5 8         3 6 2 7 4 5 1 8         8 1 5 4 7 2 6 3         8 1 5 4 7 2 6 3

Bundle:                 Corresponding MOLS:
1 3 8 6 7 5 2 4         1 8 4 5 2 7 3 6
7 5 2 4 1 3 8 6         3 6 2 7 4 5 1 8
2 4 7 5 8 6 1 3         8 1 5 4 7 2 6 3
8 6 1 3 2 4 7 5         6 3 7 2 5 4 8 1
6 8 3 1 4 2 5 7         7 2 6 3 8 1 5 4
4 2 5 7 6 8 3 1         5 4 8 1 6 3 7 2
5 7 4 2 3 1 6 8         2 7 3 6 1 8 4 5
3 1 6 8 5 7 4 2         4 5 1 8 3 6 2 7
```

Figure 16: Example transform from $\Pi\Sigma$ to M

.

5.6.3 Transforming MOLS to reduced MOLS

To show that these MOLS are indeed mutually orthogonal, we now would like to define another transformation which is a bijective mapping from the MOLS of M to another set $M_{reduced}$. We make use of an important theorem of Vanpoucke (Theorem 13) on the orthogonality of MOLS and their transforms.

Theorem 13. *For a set of MOLS, a random permutation on the alphabet of the Latin squares does not affect the orthogonality of those Latin squares. [8]*

Applying a permutation on an orthogonal set of Latin squares does not affect the orthogonality. We therefore define the mapping of the first row of all M, which by Theorem 10 above is the permutation σ^{-1}, to map to σ. We may permute the alphabet of M with the permutation σ as per the following transform.

$$T2 : M \to M_{reduced} := \{M_{reduced}(i,j) = k | M_{ij} = \sigma^{-1}(k)\} \tag{1}$$

It can easily be seen that $T2$ simply finds the elements in $M_{ij} = m$ given by $\sigma^{-1}(k) = m$ and replaces them with their index k. This is akin to the transformation directly on some $M(k,j) : M_{reduced}(k,j) = \sigma(M(k,j))$. We now show the important properties of these reduced Latin squares which allow us to draw the conclusion that these squares are orthogonal, thus proving the 1-intersecting property of the bundles in A.

Theorem 14. *The first element of $M_{reduced}$, $M^1_{reduced}$, is a symmetric reduced Latin square.*

Proof. By Theorem 11, M^1 is symmetric, with $M^1_{ij} = M^1_{ji}\ \forall\ (i,j)$. Then $T2(M^1) = M^1_{reduced}$ where $M_{reduced}(i,j) = k | M_{ij} = \sigma^{-1}(k)$. If $M^1_{ij} = \sigma^{-1}(k)$, then $M^1_{ji} = \sigma^{-1}(k)$, and $M_{reduced}(j,i) = k = M_{reduced}(i,j)$. Furthermore, by Lemma 11.1 we have that all column elements in M^1_{i1} were given by $\sigma(i)$. Thus all column elements in $M^1_{reduced}$ are now $M^1_{reduced}(i,1) = i$. Thus $M^1_{reduced}$ is a reduced Latin square. □

Theorem 15. *The kth row in $M^N_{reduced}$ is given by π_i where $i = \sigma^{-N}(k)$.*

Proof. Let the original 1-intersecting bundles be defined $A(i,j) = k$. Then consider the corresponding reduced MOLS by the two transforms we have defined. First, let $M_{reduced}(k,j) = l = \sigma(M(k,j))$. Then recall by T, $M(k,j) = i$. So $M_{reduced}(k,j) = \sigma(i)$ is the compounded expression $M_{reduced} = T2(T^{-1}(A))$. The value of k for a given bundle A^N is $(\pi_i\sigma^N)(j)$. The value of index $M_{reduced}((\pi_i\sigma^N)(j), j) = \sigma(i)$. We know $\pi_i(j) = k \iff \pi_i(k) = j$ since $\pi_i = \pi_i^{-1} \forall\ \pi_i$. We would like to show $M_{reduced}(k,j) = \pi_{\sigma^{-N}(k)}(j)$. So we must show $\sigma(i) = \pi_{\sigma^{-N}(k)}(j)$. The inverse of σ^N is σ^{r-1-N}, which is σ^{-N}. Then $\sigma = \sigma^{1-N}\sigma N$. We can express the first element in any row which is the result of applying π_i on σ^N as $\pi_i\sigma^N(j) = \sigma^{N+1}(i)$. Applying the inverse property, $\pi_i\sigma^{N+1}(i) = \sigma^N(j)$, and $\pi_i\sigma(i) = j$ for the first group. Now we consider the index $i = \sigma^{-N}(k)$. $\pi_{\sigma^{-N}(k)}\sigma(\sigma^{-N}(k)) = j$ and we can apply the inverse property yet again to yield $\pi_{\sigma^{-N}(k)}(j) = \sigma(\sigma^{-N}(k))$. Finally, $\sigma(\sigma^{-N}(k)) = \sigma^{(1-N)}(k)$ and $k = \sigma^N(i)$. Distributing through, we have $\pi_{\sigma^{-N}(k)}(j) = \sigma(i)$. Thus proven.

□

Each row in a square in M is given by $\pi_{(\sigma^{-N}(i))}(\sigma^{-1})$. Since we simply replace $\sigma^{-1}(j)$ with j, each row for the Nth Latin square $M^N_{reduced}$ is given by $\pi_{(\sigma^{-N}(i))}$. These conjectures are demonstrated in our example case for $(r-1) = 8$:

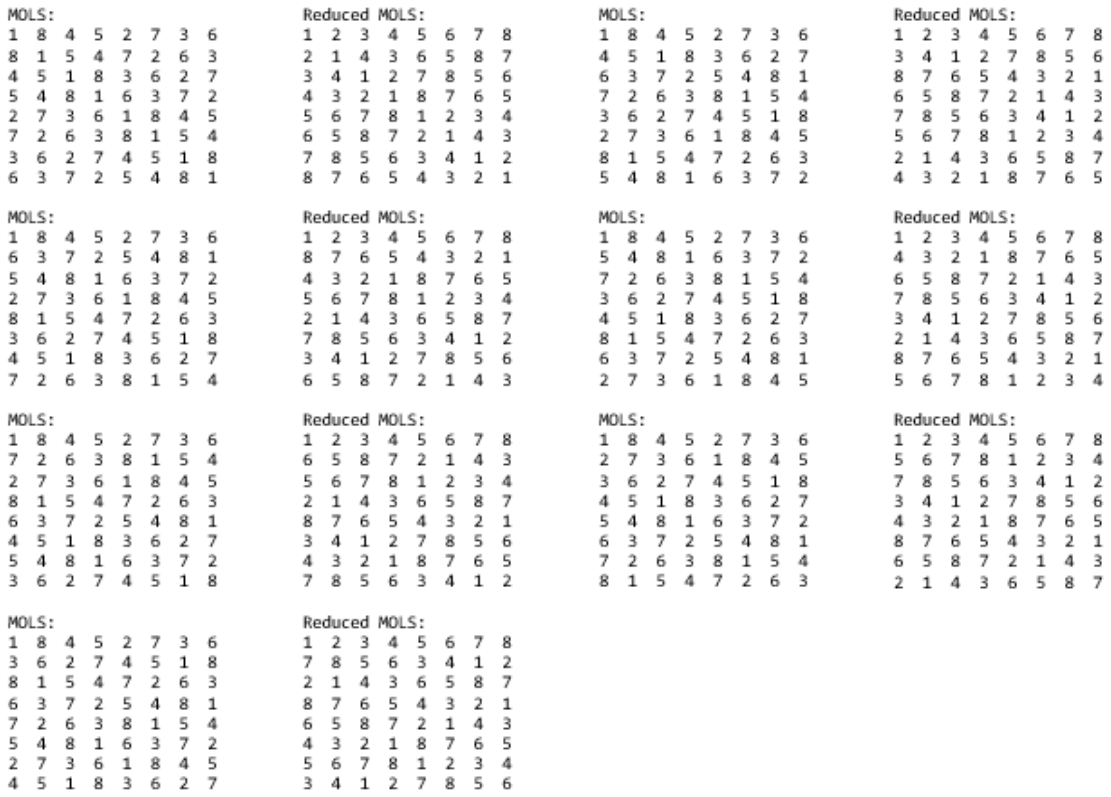


Figure 17: Example transform from M to $M_{reduced}$

.

Theorem 16. *Let τ_i and τ_j be two automorphisms. Two reduced squares $Mi_{reduced}$ and $Mj_{reduced}$ will be orthogonal provided that the automorphisms τ_i and τ_j have the property that $u(\tau_i) \neq u(\tau_j)$ for any element u other than the identity in $\{I, U2, U3, ..., U(r-2)\}$, some group of $(r-2)$ permutations keeping one symbol fixed. [12]*

Corollary 16.1. *All pairs $M^N_{reduced}$, $M^{N'}_{reduced}$ are mutually orthogonal.*

Proof. We must show each row of each square in $M^N_{reduced}$ is given by a τ_i st $M^N(\tau_i) \neq M^N(\tau_j)$. Let $M^N = u$ and $\pi_i = \tau_i$. In fact no row i in $M^N_{reduced}$ given by $\pi_{(\sigma^{-N}(i))}$ will have any elements in common with any other row j in $M^N_{reduced}$ besides the identity. Thus every $M^N_{reduced}$ is a valid Latin square. For some other $M^{N'}_{reduced}$, $r = (\sigma^{-N}(i) \neq \sigma^{-N'}(i) = r'$. Since each $M^N_{reduced}$ is a permutation on Π, the row π_r must appear at some $j \neq i$ in $M^{N'}_{reduced}$. No π_i share any elements with any other π_i, so $u(\tau_i) \neq u(\tau_j)$ for any $M^N_{reduced}$. Then $M^{N'}_{reduced}$ share one row in common, I, and for all other $(i,j), i = 2...(r-1), j = 1...(r-1)$, $M^N_{reduced}(i,j) \neq M^{N'}_{reduced}(i,j)$. □

Theorem 17. *For any $M^N_{reduced}$ the first element in a row i (which is equivalent to the column) gives the positions of the element 1 in the row i; that is, $M^N_{reduced}(i,1) = k \implies M^N_{reduced}(i,k) = 1$.*

Proof. This simply follows from the fact that each row is a permutation π_i. As per the assumption in Conjecture 22, all π_i have the property that $\pi_i(j) = k \iff \pi_i(k) = j$. □

We put the MOLS in reduced form by applying $T2(M^N) = M^N_{reduced}$, and we show these reduced form Latin squares are indeed mutually orthogonal. Since we have shown that we constructed a set of mutually orthogonal Latin squares $M_{reduced}$, M is also a set of $(r-2)$ MOLS and the original 1-intersecting bundles

are valid 1-intersecting bundles. However, we can still perform one more step to generate another set of 1-intersecting bundles, which have the symmetric property.

5.6.4 Casing reduced MOLS to 1-intersecting bundles

We re-apply the transform T to the set $M_{reduced}$ to yield a new set of 1-intersecting bundles. We show several key properties of this new set:

Theorem 18. *For the first square in the set,*$M^1_{reduced}$, $M^1_{reduced}(i,j) = k \iff M^1_{reduced}(k,i) = j$.

Proof. $M^1_{reduced}$ has $M^1_{reduced}(j,j) = 1$; $M^1_{reduced}(1,j) = j$ and $M^1_{reduced}(i,1) = i$. In general, the location j of the elements k in the ith row is given by the element $M^1_{reduced}(k,i)$. That is, $M^1_{reduced}(i,j) = k \iff M^1_{reduced}(k,i) = j$. □

Corollary 18.1. *If* $M^1_{reduced}$ *is a reduced form Latin square,* $A^1_{reduced} = T(M^1_{reduced}) = M^1_{reduced}$.

Proof. By the symmetry of $M^1_{reduced}, M^1_{reduced}(k,i) = M^1_{reduced}(i,k) = j$ Translating to a, $A^1_{reduced}(k,j) = i \iff A^1_{reduced}(j,i) = k = A^1_{reduced}(i,j)$. $M^1_{reduced}(i,j) = M^1_{reduced}(j,i) = k$. Then $A^1_{reduced}(i,j) = A^1_{reduced}(j,i)$ and $A^1_{reduced}(j,j) = 1$. □

Theorem 19. $A^N_{reduced} = T(M^N_{reduced})$ *for any* $M^N_{reduced} \in M_{reduced} \implies A^N(i,i) = 1$.

Proof. Because all $M^N_{reduced} \in M_{reduced}$ have a first row given by I, by definition of reduced, we can write the elements $M^N_{reduced}(1,j) = j$. Then by the transform, $A^N_{reduced}(j,j) = 1$. □

Corollary 19.1. $A^N_{reduced} = T(M^N_{reduced})$ *for any* $M^N_{reduced} \in M_{reduced} \implies$ *the first row (and column) of* $A^N_{reduced}$ *is given by* σ^{N-2}.

In the example in Figure 18, the final 1-intersecting bundles which correspond to the reduced-form MOLS are permutations on the original σ_i values under the normalized symbol set. Generally, the theorems above can be summarized in the following observations:

- Since $\pi_i(i) = 1\ \forall i$, $\sigma_i(1) = 1\ \forall \sigma$, and $\sigma(1) = 1$, the diagonal of every bundle is 1. This holds for a bundle of any MOLS in reduced form.
- Similarly, $\pi_i(1) = \sigma(i)$, meaning the first row is mirrored as the first column; this also holds for any MOLS in reduced form.
- Finally, $\pi_i(j) = k\ \forall (j,k) \neq i$, so $A_{ij} = A_{ji} = k$ and A is in fact a symmetric Latin square with $A = A^T$.

The final normalized MOLS and corresponding 1-intersecting bundles under our example are:

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7	2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6	3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3	6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2	7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
3	4	1	2	7	8	5	6	3	4	1	2	7	8	5	6
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
6	5	8	7	2	1	4	3	6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2	7	8	5	6	3	4	1	2
2	1	4	3	6	5	8	7	2	1	4	3	6	5	8	7
5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4
4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	7	2	8	6	4	5	3
3	4	1	2	7	8	5	6	7	1	8	2	4	6	3	5
8	7	6	5	4	3	2	1	2	8	1	7	5	3	6	4
6	5	8	7	2	1	4	3	8	2	7	1	3	5	4	6
7	8	5	6	3	4	1	2	6	4	5	3	1	7	2	8
5	6	7	8	1	2	3	4	4	6	3	5	7	1	8	2
2	1	4	3	6	5	8	7	5	3	6	4	2	8	1	7
4	3	2	1	8	7	6	5	3	5	4	6	8	2	7	1

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	5	7	3	4	8	6	2
8	7	6	5	4	3	2	1	5	1	3	7	8	4	2	6
4	3	2	1	8	7	6	5	7	3	1	5	6	2	4	8
5	6	7	8	1	2	3	4	3	7	5	1	2	6	8	4
2	1	4	3	6	5	8	7	4	8	6	2	1	5	7	3
7	8	5	6	3	4	1	2	8	4	2	6	5	1	3	7
3	4	1	2	7	8	5	6	6	2	4	8	7	3	1	5
6	5	8	7	2	1	4	3	2	6	8	4	3	7	5	1

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	6	5	2	8	3	4	7
4	3	2	1	8	7	6	5	6	1	2	5	3	8	7	4
6	5	8	7	2	1	4	3	5	2	1	6	4	7	8	3
7	8	5	6	3	4	1	2	2	5	6	1	7	4	3	8
3	4	1	2	7	8	5	6	8	3	4	7	1	6	5	2
2	1	4	3	6	5	8	7	3	8	7	4	6	1	2	5
8	7	6	5	4	3	2	1	4	7	8	3	5	2	1	6
5	6	7	8	1	2	3	4	7	4	3	8	2	5	6	1

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	4	6	7	3	2	8	5
6	5	8	7	2	1	4	3	4	1	7	6	2	3	5	8
5	6	7	8	1	2	3	4	6	7	1	4	8	5	3	2
2	1	4	3	6	5	8	7	7	6	4	1	5	8	2	3
8	7	6	5	4	3	2	1	3	2	8	5	1	4	6	7
3	4	1	2	7	8	5	6	2	3	5	8	4	1	7	6
4	3	2	1	8	7	6	5	8	5	3	2	6	7	1	4
7	8	5	6	3	4	1	2	5	8	2	3	7	6	4	1

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	8	4	5	2	7	3	6
8	1	5	4	7	2	6	3	8	1	5	4	7	2	6	3
4	5	1	8	3	6	2	7	4	5	1	8	3	6	2	7
5	4	8	1	6	3	7	2	5	4	8	1	6	3	7	2
2	7	3	6	1	8	4	5	2	7	3	6	1	8	4	5
7	2	6	3	8	1	5	4	7	2	6	3	8	1	5	4
3	6	2	7	4	5	1	8	3	6	2	7	4	5	1	8
6	3	7	2	5	4	8	1	6	3	7	2	5	4	8	1

MOLS:								Corresponding Bundle:							
1	2	3	4	5	6	7	8	1	3	8	6	7	5	2	4
7	8	5	6	3	4	1	2	3	1	6	8	5	7	4	2
2	1	4	3	6	5	8	7	8	6	1	3	2	4	7	5
8	7	6	5	4	3	2	1	6	8	3	1	4	2	5	7
6	5	8	7	2	1	4	3	7	5	2	4	1	3	8	6
4	3	2	1	8	7	6	5	5	7	4	2	3	1	6	8
5	6	7	8	1	2	3	4	2	4	7	5	8	6	1	3
3	4	1	2	7	8	5	6	4	2	5	7	6	8	3	1

Figure 18: Reduced-form MOLS and corresponding bundles from novel construction, $(r-1)=8$

5.7 Vertex cover in the maximally sized 1-intersecting graph

We now return to the original problem. In the maximally sized set of 1-intersecting bundles, every edge intersects every other edge in every other bundle exactly once. Thus the vertices contained in any one edge are a valid cover (with r edges), and in fact the vertices contained in any one part are a valid cover. Since we have shown there are $r-1$ vertices in every part, taking any part to be the cover gives a valid $(r-1)$-sized vertex cover.

Theorem 20. *In the maximally sized 1-intersecting hypergraph, every vertex has degree $r-1$.*

Proof. We have two cases here: either a vertex is in some 1-intersecting bundle, or it is the vertex in which the 1-intersecting bundle is incident upon. If the latter, then the $(r-1)$ edges in the 1-intersecting bundle are the only edges incident upon that vertex by definition, and the vertex has degree $(r-1)$. Now consider any vertex in a 1-intersecting bundle. We know any edge in a bundle necessarily intersects the $(r-1)$ edges in $(r-2)$ other bundles. Additionally it may not intersect any of these other edges more than once. There are $(r-1)$ possible places to intersect, and $(r-1)(r-2)$ edges to intersect. Thus exactly $(r-2)$ edges must intersect the first at at each vertex, and therefore the degree of that vertex is $(r-1)$. □

In fact in the maximally sized 1-intersecting hypergraph, all edges have degree $d_v = (r-1)$. We next show that if all edges have $d_v = (r-1)$, the part size must be $(r-1)$.

Theorem 21. *If a hypergraph is $(r-1)$-regular and r-partite, then the size of every $V_i \in V$ is $(r-1)$.*

Proof. This is a simple proof by counting over the edges and over the parts. Summing over the vertex degrees in the edges, $\sum_v d_v = \binom{r-1}{2} = rm$. Since $m = (r-1)^2$ in the prime power case, the sum over degrees is $r(r-1)^2$. In another way of counting, we have r parts, x vertices per part, and d_v degree of each vertex. There are r parts and $r-1$ is the degree of the vertex. Therefore $r(r-1)(x) = r(r-1)^2 \implies x = r-1$. □

We conclude that we may select $(r-1)$ vertices from the same part which yield a valid vertex cover, therefore proving Ryser in this very limited setting.

5.8 Other areas of note

5.8.1 Maximizing 1-intersecting hypergraph for non prime-power orders

Extensive research has gone into the construction of MOLS for non-prime powers. MOLS for non-primes have different constructions from primes and prime powers. Perhaps the most useful is the Kronecker Product of two Latin Squares of order m and n respectively. This will yield an $mn \times mn$ square whose symbols given as ordered pairs of the smaller array can be assigned to the natural ordering of symbols to yield a Latin square. Then we have

$$N(mn) \geq \min(N(m), N(n))$$

[27].

In fact no pair of orthogonal Latin squares exists for the order $n = 6$ [24]. Does this mean for a 7-partite hypergraph it is impossible to achieve even $3(r-1)$ edges? The case seems improbable, but is certainly an open question for more exploration.

5.8.2 Applications of MOLS

One interesting application of MOLS is the problem of creating error-correcting codes, used to control errors in noisy data over telecommunication channels. Tang [27] shows via construction the equivalence between the existence of a number of MOLS and the existence of error-correcting codes of a q-ary q^2 code.

The construction we pose seems to be related to the primitive root construction for applications in other signal processing domains. For example, the Fast Fourier transform generates its pairings using reverse-order readings of the binary digits of $1...N$ samples, hence applying a similar operation in a binary pairing but lacking the additional level of negation.

Another important application is in cryptography; Latin squares, and MOLS, can be used in the creation of codes. A message of length m can be encoded via MOLS. In particular, only 2 MOLS of order m are needed to encrypt a message of length m. The only impossible message lengths are therefore $m = 2$ and $m = 6$ [**?**].

5.8.3 Cyclic permutation algorithm for other prime powers

It was conjectured in Section 5.6.1 that this process may be replicated for other prime powers. In fact we empirically find that the process does work for powers of 3, and include some of the constructions here for interest. The major difference in the choice of the powers of 3 is that the strictly transitive permutation on 3 is (3 1 2) instead of (2 1) and therefore the construction of the permutation matrix Π depends on a local 3-cycle instead of $\pi_i(j) = k \iff \pi_i(k) = j$. The permutations π_i are therefore no longer their own inverse, but their own cyclic multiples, $\pi_i(\pi_i(\pi_i) = \pi_i^3 = I$.

Sub-groups therefore follow the pattern in the $r-1=9$ case: (1 2 3), (3 1 2), (2 3 1). When the elements switch locally, they follow the same pattern. In Figure 19, following the triplet (1 2 3) shows it first in triple position 1, then 2, then 3; then the elements permute locally to (3 1 2) and follow the pattern in the triple positions (3 1 2); and so forth for the final 3 positions, (2 3 1).

Figure 19: Example of construction of Π for $r-1=9$

We include the reduced MOLS and symmetric 1-intersecting bundles for $r-1=9$ in the interest of illustrating the differences between the strictly transitive permutations on 3 and 2. For $p=3$, if $seq[i]=3$ and C_p is (312), $C_p^{-1}=(231)$ then π_1, the identity [1 2 3 4 5 6 7 8 9] would be permuted to $\rightsquigarrow$ [7 8 9 1 2 3 4 5 6]. Applying the same permutation to the next row yields $\rightsquigarrow$ [4 5 6 7 8 9 1 2 3]. Then, as seq(3) = 1, we now apply the permutation at the local level, [4 5 6 7 8 9 1 2 3] $\rightsquigarrow$ [6 4 5 9 7 8 3 1 2]. This is equivalent to applying C_p^{-1} to I at the 3 level and then C_p at the 1-level.

Doing so yields $\sigma=$ [1 7 4 6 3 9 8 6 2]. We then may construct $(r-2)=8$ mutually orthogonal Latin squares from the cyclic permutation approach.

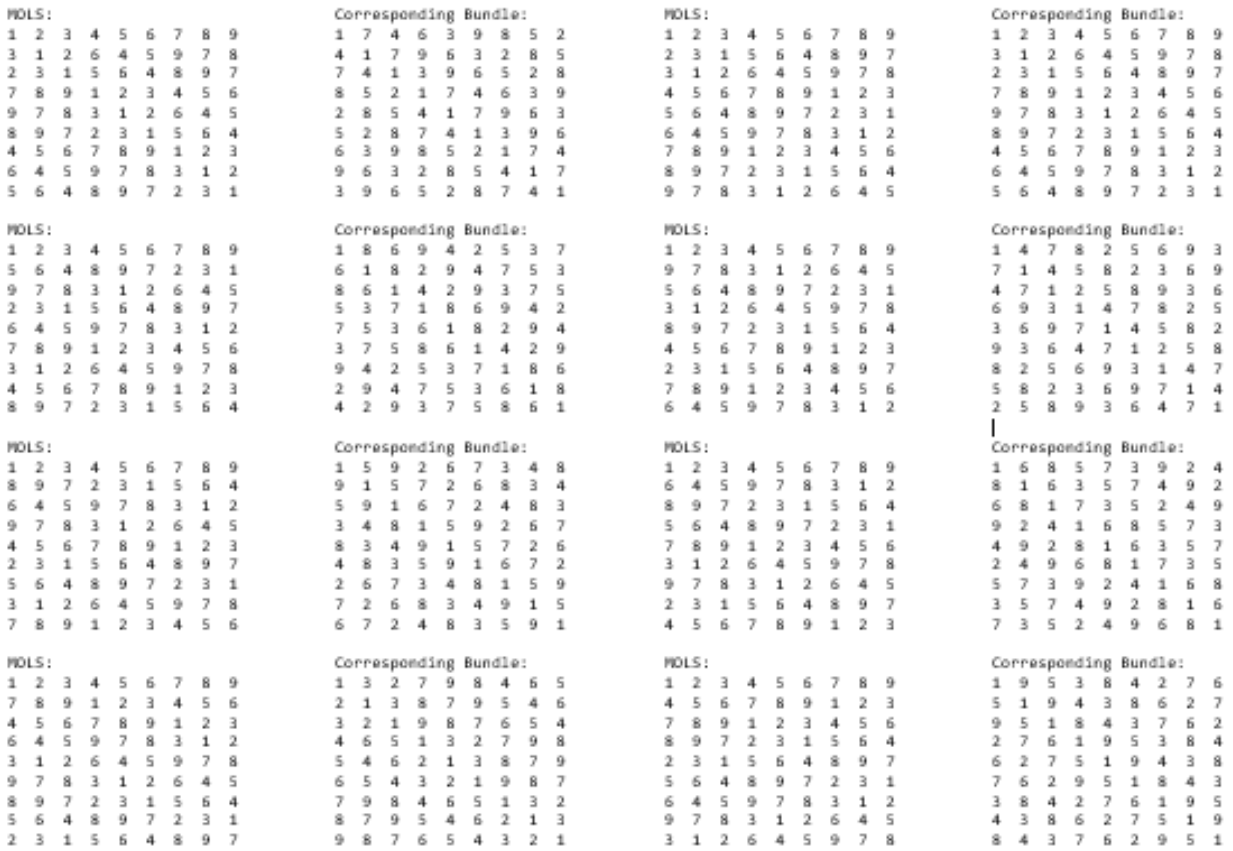


Figure 20: Reduced-form MOLS and corresponding bundles from novel construction, $(r-1) = 8$

We conjecture the following.

Conjecture 22. *If $(r-1)$ is a prime power, expressed $(r-1) = p^d$, such that $pd+1$ is also a prime power with $pd+1 = p'^{d'}$ such that $p'd'+1$ is a prime power, and recursively so forth, then the result returned by the cyclic permutation algorithm Π and σ have the property that all $\pi_i^p = I$, and the element 1 is fixed in σ with the subsequent $r-2$ elements forming an $(r-2)$ orbital.*

As of writing, we have no concrete proof of Conjecture 22. However, we observe the following patterns empirically. Note that some powers of 3 work because they have a sharply transitive permutation as well. We verify empirically using Java that the construction works for certain prime powers, as shown in Table 9.

$r-1$	prime power	succeeds
3	3	Y
4	2^2	Y
5	5	N
8	2^3	Y
9	3^2	Y
16	2^4	N
25	5^2	N
27	3^3	N
32	2^5	Y
64	2^6	Y
81	3^4	N
128	2^7	N
243	3^5	Y
256	2^8	N
512	2^9	N
1024	2^{10}	Y

Table 9: Prime powers for which the cyclic permutation construction empirically succeeds

5.8.4 Implications for Ryser

Lovász conjectures that the inequality in Ryser's Conjecture cannot be improved when $(r-1)$ is a prime power [1]. This conjecture poses an interesting overlap between the conditions under which maximally sized 1-intersecting hypergraphs are possible and the conditions under which the maximally sized hypergraphs in general arise.

Our findings confirm this conjecture since we are able to show that a maximally sized hypergraph is possible if $(r-1)$ is a prime power, $|M| = 1$, and $k = 1$. theorize this could be due to an optimal distribution of vertices which is possible under these conditions, maximizing the size of the 1-intersecting hypergraph, and thereby making it quite difficult to find a good candidate vertex cover even for general hypergraphs. In the k-intersecting hypergraph the problem is even less constrained; and when the intersection size is not defined, this is $|M| = 1$. There may be some application of the intuition of optimally spreading out edges which leads to more difficult vertex covers.

6 Code

Code related to this effort has been published on GitHub at
`https://github.com/annadodson787/Honors-Thesis---Ryser-for-Intersecting-Hypergraphs`.

In the repo, four main-method files comprise the algorithmic approach:

- PartitionedGraphDriver.java
- PermutationCycle.java
- MOLSGenerator.java
- Edge Score Visualizer.iypnb

The code related to this the $|M| = 1$ reactive hypergraph and the 1-intersecting reactive hypergraph (empirical findings) is available in the file PartitionedGraphDriver.java. The generative algorithm for the 1-intersecting bundles is found in PermutationCycle.java. The generative algorithm which uses the traditional

finite field generative algorithm and casts the resulting MOLS to a 1-intersecting bundle is found in MOLS-Generator.java. The metrics and visualizations are found in Edge Score Visualizer.ipynb.

6.0.1 PartitionedGraphDriver.java

PartitionedGraphDriver.java is the main file for the reactive hypergraph and is structured in several classes and objects. It must represent a hypergraph, generate vertex covers and edges as per the algorithms outlined above, and drive the game between Alice and Bob.

The hypergraph is represented as Vertex and Edge objects. When creating a partitioned hypergraph, the number of parts is specified (N in the code), and the size of the maximal matching is specified (M in the code). Each Edge therefore has degree N. It is represented as an array of Vertex objects. Each Vertex object has a PartName and an integer value. The PartName is an enumeration of parts $v_{11}....v_{1N}$, where N is the number of parts in the hypergraph. The value i is the ith vertex in a given part. Edges also have associated metric counters for visualizations that must be set on them by the driving methods.

The hypergraph itself has several properties as well. It has a order (set by N), a set of Edges, and several mappings. The *vertexToEdgeMap* is a mapping of all vertices in the hypergraph to edges they participate in; that is, edges they are incident on. The $partToVertexMap$ is a map of a given part to all vertices in that part. The $stringToVertexMap$ is a quick way to lookup vertices for the CLI feature. A hypergraph also has associated strings for building out the game. Since the games are generated recursively in many cases, we manage the string objects to print out the games when they terminate.

The driving methods are four in number:

1. playGreedyGame - back-and-forth between Alice and Bob, playing with the vertex cover algorithm and greedy reactive edges.
2. playBruteForceGameRecursive - recursively explores every possible move that Bob could generate at a given step, with options to recurse on all edges, all greedy edges or all 1-overlap edges.

   playBruteForceGameRecursiveHelper

   playBruteForceGameOverlapRecursiveHelper

   recursiveGreedyCandidateHelper
3. playCLIGame - allows user to specify the edge they wish to play. The game verifies that the edge is valid and reacts accordingly, adding the edge to the hypergraph and playing a new vertex cover.
4. (deprecated) playBruteForceGame - uses a FIFO queue to create a new hypergraph object for every game; extremely computationally costly, but works the same as the recursive method without needing to store the game strings.

The algorithms for the vertex cover include 2 different approaches, one which removes all edges already hit and the other which uses just those vertices of maximum participates.

1. aliceAlgorithm - the algorithm for choosing a vertex cover which does not update the priority queue, choosing the $(r-1)$ vertices with top maximal degree in the hypergraph.
2. greedyAlice - the same, but removing the edges incident on each vertex chosen and updating a set edgesLeftToHit

The PartitionedGraphRestructured keeps track of the current state of a hypergraph with the following properties: order, depth, and several scoring metrics. It maintains a set of edges in the hypergraph, and the mappings of vertices to incident edges, parts to vertices in the part, and strings to vertices (for easy lookup). It also builds out a string that represents a transcript of a given hypergraph which terminates when there

are no possible edges to play in reaction to a vertex cover choice. PartitionedGraphRestructured objects can be initialized from scratch by specifying order, or from an existing hypergraph, in which case all objects in the old hypergraph are cloned into new parts in memory to avoid issues with pointer modification.

The algorithms for generating the reactive hypergraph include several: one for generating a single greedy edge as per the algorithm, one which generates all greedy edges, one which generates all edges, and one which generates all edges that satisfy the 1-overlapping property.

The methods available within the PartitionedGraphRestructured as are follows:

1. addEdge - adds a new edge to the PartitionedGraphRestructured.
2. removeEdge - removes an existing edge from the PartitionedGraphRestructured.
3. generateAllMoves - generate all possible moves in response to a VC choice and the current state of the hypergraph.

   addAllPossibleEdges - helper method to recursively get the set of possible edges.
4. generateAllGreedyMoves - generate all possible moves which are considered 'greedy' (i.e. maximize edge participation in the vertex choices) in response to a VC choice and the current state of the hypergraph.

   addAllGreedyEdges - helper method to recursively get the set of greedy edges.
5. generateOverlapMoves - generate all moves which overlap every other edge just once in response to a VC choice and the current state of the hypergraph

   addAllOverlapEdges - helper method to recursively get the set of overlap edges.
6. generateMove - generates a single move as per the greedy algorithm in response to a vertex cover.
7. stringRepresentation - represent the hypergraph as a string; this gives the 'game transcript' which can be found in this document and in its supporting files.
8. generateCountString - helper method to represent the degrees of vertices in the hypergraph at a given time step.
9. scoreCandidateEdge - explore ways of scoring edges as per greediness or other metrics.

   scoreCandidateHelper - helper for the score method.
10. candidateEdgeIsGreedy - returns whether a candidate edge choice is greedy, taking any possible ordering of the vertices in it.

    recursiveGreedyCandidateHelper - helper method for greediness evaluation.

The class also includes a number of methods for formatting, scoring, and generating metrics (.csv files) for further parsing.

6.0.2 PermutationCycle.java

The class PermutationCycle.java generates maximal sets of 1-intersecting bundles for prime power r-partite hypergraphs which take the form $r-1 = p^d$ st $pd+1$ is a prime power using the cyclic permutation algorithm. It illustrates the construction of the $(r-1)$ bundles, verifies them, casts them to MOLS, reduces the MOLS, and casts them back to 1-intersecting bundles with the property that all bundles intersect with all other bundles on the diagonal.
Important methods include:

1. generateDivisorSequence - generate the divisor sequence as per the cyclic permutation algorithm.

2. generatePermutations - generate the matrix Π based upon the cyclic permutation algorithm.
3. generatePermutationSet(deprecated) - places all the permutations into a set for lookup, then creates a matrix based on a seed order
4. generateBundle - generate a 1-intersecting bundle given a seed (akin to σ) and a permutation matrix Π.
5. applyPermutation - helper method for applying a permutation.
6. getCol1 - returns the first column of a $(r-1) \times (r-1)$ matrix, which may be used as the seed for the following matrix.
7. getDiag (deprecated) - returns the diagonal of a $(r-1) \times (r-1)$ matrix, which may be used as the seed for the following matrix.
8. addAllRows - adds all rows to a set for lookup
9. checkRowIsFirstRow - quickly checks whether this bundle has been seen before; i.e., if we have reached a cycle before $(r-2)$ iterations and thus the cyclic decomposition of σ is less than $(r-2)$.
10. checkRow - checks a row against all previous rows and ensures it 1-intersects all previous rows.
11. isSameAsPreviousRow - returns whether or not a new row is identical to a previous row
12. MOLSto1Intersecting - Applies the transform T.

    MOLSto1Intersecting - Applies the transform, in the other direction.
13. rearrangeSymbols - helper method to reassign the symbols of a matrix according to the natural elements $1...(r-1)$ given a seed array and replacing all instances of the ith element in the array with i.
14. inverse - helper method to compute the inverse of a permutation.

The output of running PermutationCycle.java with an input prime power $q = p^d$ is as follows:

- If $q = p^d$ is not of the form such that $pd + 1$ is a prime power, then the program will terminate early and will first print the generator sequence, followed by the length of the cycle in the permutation that is generated by the sequence.
- If $q = p^d$ is of the form $pd + 1$ a prime power, the program will generate a set of $(q-1)$ MOLS and 1-intersecting bundle pairs. It will first print the generator sequence. It will then print the (r-1) bundles, followed by the (r-1) the MOLS in non-reduced form, convert them, and print them in reduced form with the corresponding 1-intersecting bundles with 1 on the diagonal.

6.0.3 MOLSGenerator.java

The MOLSGenerator uses the finite field approach to MOLS construction for prime powers of the form $q = p^d$, casts MOLS to reduced MOLS to 1-intersecting bundles, and checks the validity of those 1-intersecting bundles. It currently uses a hard-coded bit representation of the primitive polynomial where each bit represents whether the term x^b is present, for $b = 0$ is the LSB and $b = d - 1$ is the MSB. As such, several methods are overlapping with PermutationCycle.java.

1. GFAdd - generates the Galois Field addition table for a given primitive polynomial and prime power.
2. GFMult - generates the Galois Field multiplication table for a given primitive polynomial and prime power.
3. getDegree - helper to get the degree of the primitive polynomial that was chosen as the seed.

4. MOLSto1Intersecting - as above.

5. addAllRows - as above.

6. checkRow - as above.

7. isSameAsPreviousRow - as above.

The output of running MOLSGenerator.java with an input prime power of 2 q is a set of $(q-1)$ MOLS and 1-intersecting bundle pairs.

6.0.4 Edge Score Visualizer.iypnb

The Python file Edge Score Visualizer.iypnb is a visualization tool which allows breakdown of hypergraphs in logfiles by their size. The program hypergraphs for each game the metric of choosing for each of Bob's plays and juxtaposes them on top of each other based on the size of the hypergraph.

The program relies on the Anaconda3+Jupyter+Python environment and uses Pandas to read .csvs into a dataframe for further handling.

6.0.5 Supporting Documents

Several logfiles are generated by the PartitionedGraphDriver to empirically view the greediness and recursions of the generative algorithms in the reactive hypergraph. These files are included in the GitHub repo, which can be found here: `https://github.com/annadodson787/Honors-Thesis---Ryser-for-Intersecting-Hypergraphs`.

The Jupyter notebook and corresponding python code in EdgeScoreVisualizer.iypnb uses these metrics to generate the figures above, among others, using Pandas dataframes along with the matplotlib library. Incomplete CSV data are read in and nonetypes removed to generate dataframes for each object.

Each line in the supporting data files represents a game. Each step represents a metric associated with one of Bob's edge choices. The supporting data files (for the $r = 4$ case, in the repo):

- depths.csv: represents the number of playable edges at each depth in the hypergraph.
- gamescores.csv: uses a scoring metric which determines how many prior edges are hit by one of Bob's edge choices at each step in the brute-force hypergraph.
- greedy.csv: represents whether or not Bob's edge choice was greedy at a given step.
- moves.csv: string representations of all of Bob's moves.
- options.csv: string representations of all of Bob's possible moves.

These files are used in the script EdgeScoreVisualizer.iypnb.

Also included in the GitHub repo: the slide deck from the Computer Science honors presentation, and a supplementary PDF of the game script for all of the greedy games in the $r = 4$ case, for further interest.

Acknowledgements

Acknowledgments are due to all of the professors and teachers along my undergraduate journey who inspired my eventual completion of this senior thesis project. A major thank-you to Deeparnab Chakrabarty, Assistant Professor of Computer Science at Dartmouth College and the advisor on this project, for his eternal commitment to the pursuit of knowledge and willingness to explore new horizons. This project would never have been possible without you.

Thanks to Peter Winkler, Department of Mathematics, for his key insight which unlocked a whole new realm of exploration in drawing the connection between the 1-intersecting hypergraph and the set of mutually orthogonal Latin squares. Without him, the project would have taken a very different form.

Finally, a special mention to Thomas Cormen, Department Chair of Computer Science at Dartmouth, who inspired my love for algorithms and who embodies the true art of making an argument, who will be stepping back after 27 years of service at the end of this academic year, and who will be missed.

References

[1] L. Lovász, "Maximum degree and fractional matchings in uniform hypergraphs," *Combinatorica*, vol. 260, no. 1, pp. 209–264, 1981.

[2] Lloyd, E. K. Biggs, and R. J. Wilson, *Graph Theory*. Oxford University Press, 1976.

[3] J. Henderson, *Permutation Decompositions of (0, 1)-matrices and decomposition transversals*. PhD thesis, 1971.

[4] V. Guruswami, S. Sachdeva, and R. Saket, "Inapproximability of minimum vertex cover on k-uniform k-partite hypergraphs," *SIAM Journal on Discrete Mathematics*, vol. 29, no. 1, p. 36–58, 2015.

[5] "Maximum bipartite matching," May 2019.

[6] R. Aharoni, "Rysers conjecture for tripartite 3-graphs," *Combinatorica*, vol. 21, p. 1–4, Jan 2001.

[7] P. Winkler, May 2020.

[8] K. H. Hicks, G. L. Mullen, L. Storme, and J. Vanpoucke, "The number of different reduced complete sets of mols corresponding to pg(2,q)," *Journal of Geometry*, vol. 109, no. 1, 2018.

[9] T. Evans, "Latin squares : new developments in the theory and applications," *Latin Squares and Universal Algebra*, pp. 203–216, 1991.

[10] J. Vanpoucke. PhD thesis, 2011.

[11] R. C. Bose, "On the application of the properties of galois fields to the construction of hyper-graeco-latin squares," *Sankhya*, vol. 3, p. 323–338, 1938.

[12] J. Dénes and A. D. Keedwall, "Latin squares - new developments in the theory and applications," *Annals of Discrete Mathematics*, vol. 46, 1991.

[13] N. Bansal and S. Khot, "Inapproximability of hypergraph vertex cover and applications to scheduling problems," *Automata, Languages and Programming Lecture Notes in Computer Science*, p. 250–261, 2010.

[14] R. R. P. Erdos, C. Ko, "Intersection theorems for systems of finite sets," *Quart. J. Math*, vol. 12, pp. 313–320, 1961.

[15] T. Bohman and R. R. Martin, "A note on g-intersecting families," *Discrete Mathematics*, vol. 260, no. 1-3, p. 183–188, 2003.

[16] V. S. A. Kumar, S. Arya, and H. Ramesh, "Hardness of set cover with intersection 1," *Automata, Languages and Programming Lecture Notes in Computer Science*, p. 624–635, 2000.

[17] E. Halperin, "Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs," *SIAM Journal on Computing*, vol. 31, no. 5, p. 1608–1623, 2002.

[18] Z. Tuza, "Ryser's conjecture on transversals of r-partite hypergraphs," *Ars Combin*, no. 16, pp. 201–209, 1983.

[19] G. J. Puleo, "Tuza's conjecture for graphs with maximum average degree less than 7," *European Journal of Combinatorics*, vol. 49, p. 134–152, 2015.

[20] P. E. Haxell and Y. Kohayakawa, "Packing and covering triangles in tripartite graphs," *Graphs and Combinatorics*, vol. 14, no. 1, p. 1–10, 1998.

[21] M. Krivelevich, "Perfect fractional matchings in random hypergraphs," *Random Structures and Algorithms*, vol. 9, no. 3, p. 317–334, 1996.

[22] J. Fox and J. Pach, "Erdős-hajnal-type results on intersection patterns of geometric objects," *Bolyai Society Mathematical Studies Horizons of Combinatorics*, p. 79–103.

[23] M. Karpinski, R. Schmied, and C. Viehmann, "Nearly tight approximation bounds for vertex cover on dense k-uniform k-partite hypergraphs," *Journal of Discrete Algorithms*, vol. 33, p. 49–57, 2015.

[24] R. C. Bose and S. S. Shrikhande, "On the falsity of eulers conjecture about the non-existence of two orthogonal latin squares of order $4t \mod 2$," *Proceedings of the National Academy of Sciences*, vol. 45, p. 734–737, Jan 1959.

[25] I. Anderson, *A First Course in Combinatorial Mathematics*. Oxford University Press, 1991.

[26] J. Westall and J. Martin, *An Introduction to Galois Fields and Reed-Solomon Coding*. PhD thesis, School of Computing, Clemson University, 2010.

[27] J. Tang, *Latin Squares and Their Applications*. PhD thesis, 2016.