REVISITING THE SPREADING AND COVERING NUMBERS

BEN BABCOCK AND ADAM VAN TUYL

Abstract. We revisit the problem of computing the spreading and covering numbers. These numbers were first introduced by Geramita, Gregory, and Roberts to study the Ideal Generation Conjecture, a conjecture concerning sets of points in a projective space. Both the spreading number and covering number can be reformulated as a graph theory question. We describe some new approaches to computing or bounding these numbers. In addition, we show a surprising connection to some of the spreading numbers to the sequence A053307 in the On-Line Encyclopedia of Integer Sequences.

1. Introduction

Let $R = k[x_1, \dots, x_n]$ be a polynomial ring over a field k. For any non-negative integer d, let M_d be the set of all monomials of degree d in R. For any subset $W \subseteq M_d$, let

$$R_1W = \{x_i m \mid m \in W \text{ and } 1 \le i \le n\},$$

that is, R_1W is the subset of M_{d+1} formed by multiplying each monomial of W by each of the variables of R. For any $W \subseteq M_d$, we always have $|R_1W| \le n|W|$ and $R_1W \subseteq M_{d+1}$.

We are interested in finding subsets W where either $|R_1W| = n|W|$ or $R_1W = M_{d+1}$. In particular, we define the **spreading number** to be

$$\alpha_n(d) = \max\{\ |W|\ :\ W \subseteq M_d \text{ and } |R_1W| = n|W|\}.$$

The terminology is derived from the fact that the elements of R_1W are "spread" out in M_{d+1}. Similarly, the **covering number** is defined to be

$$\rho_n(d+1) = \min\left\{\ |W|\ :\ W \subseteq M_d \text{ and } R_1W = M_{d+1}\right\}.$$

In this case the elements of R_1W "cover" the elements of M_{d+1}. The goal of this paper is to introduce some new approaches to computing and bounding these numbers.

The spreading and covering numbers were introduced by Geramita, Gregory, and Roberts to study the Ideal Generation Conjecture for a set of generic points in $\mathbb{P}^n$ (see [3, Theorem 4.7]). Geramita, et al. gave exact values for $\alpha_n(d)$ for all d when $n = 3$ or 4, and some scattered results and bounds on other values. Curtis [2] later found a formula for $\rho_3(d)$ for all d and an improved lower bound on $\rho_4(d)$. Using techniques from linear programming, Hulett and Will [4] improved these lower bounds on $\rho_4(d)$. Carlini, Hà, and the second author [1] later reformulated the problem to show it is equivalent to computing the dimension of some abstract simplicial complex.

1991 *Mathematics Subject Classification.* 13F20 (13F55 13P99 68W30).
Key words and phrases. monomials, edge ideals, spreading and covering numbers.
Last updated: September 27, 2011.

The computation of these invariants has proven to be quite challenging. Computing $\alpha_n(d)$ or $\rho_n(d+1)$ by brute force, even for small n, quickly proves infeasible due to the number of subsets one must check. For example, to compute $\alpha_5(5)$, we have $|M_d| = \binom{5+5-1}{5} = 126$, so there are $2^{126} \approx 8.5 \times 10^{37}$ subsets that need to checked using a naïve method. We have therefore focused on introducing algorithms that bound $\alpha_n(d)$ and $\rho_n(d+1)$. Unfortunately, as n and d increase, so does the distance between our two bounds. Given the difficulty of computing these invariants, we have therefore set ourselves a goal of bounding as many of the spreading and cover numbers in a neighbourhood of a size of no more than 100.

To find these bounds, we make use of a construction of Geramita, et. al; in particular, we make a graph $S_n(d)$ where the vertices of this graph correspond to the elements of M_d. The values of $\alpha_n(d)$ and $\rho_n(d+1)$ are then related to graph theoretic invariants of $S_n(d)$. In Section 2, we make this connection explicit, and reframe our problem in terms of $S_n(d)$. We also highlight the symmetry of $S_n(d)$, which we exploit in our algorithms.

In Section 3, we focus on an algebraic method to compute an upper bound on $\alpha_n(d)$. We associate to $S_n(d)$ its edge ideal $I(S_n(d))$. Computing $\alpha_n(d)$ then reduces to computing the Krull dimension of $T/I(S_n(d))$ where T is some suitable polynomial ring. We construct suitable linear forms $L_1, \ldots, L_s$, to make the ring $T/(I(S_n(d)) + (L_1, \ldots, L_s))$. The dimension of this new ring allows us to give an upper bound $\alpha_n(d)$.

In Section 4 we describe two greedy algorithms. The first algorithm gives a lower bound for $\alpha_n(d)$, while the second gives an upper bound for $\rho_n(d+1)$. Both exploit the symmetry of the graph $S_n(d)$. We also include refinements to both algorithms so that they can be adapted to compute $\alpha_n(d)$ and $\rho_n(d+1)$ exactly. We summarize our output in Section 5 where we compare output to the bounds found in previous papers.

In Section 6 we prove that the sequence $\alpha_4(d)$, one of the few cases for which we have a formula, equals the sequence A053307 found on the On-Line Encyclopedia of Integer Sequences [7]. This sequence counts the number of 2×2 matrices with non-negative entries that sum to d up to permutation of rows and columns. One is lead to ask whether other spreading or covering numbers are related to interesting integer sequences.

As a final note, all of the code that we used will be available on the authors' websites[1].

Acknowledgements. This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca) and Compute/Calcul Canada. We used *Macaulay 2* [6] to carry out our calculations. The first author was supported by an NSERC USRA during the summers of 2010 and 2011. The second author was supported by an NSERC Discovery Grant.

2. Preliminaries

We recall the relevant definitions and preliminary results that we will need for the remainder of the paper. As in the introduction, we let $R = k[x_1, \ldots, x_n]$ denote the

[1]`https://github.com/tachyondecay/spreading-covering-numbers/`
`http://flash.lakeheadu.ca/~avantuyl/research/SpreadCover_Babcock_VanTuyl.html`

polynomial ring over a field k, and M_d will denote the monomials of degree d in R. We first translate our problem of computing $\alpha_n(d)$ and $\rho_n(d+1)$ into a graph theory problem.

Construction 2.1. Fix positive integers n and d. Let $S_n(d)$ denote the graph whose vertex set is the set of monomials M_d in $R = k[x_1, \ldots, x_n]$, and two vertices m_i, m_j are adjacent if and only if $\deg\big(\operatorname{lcm}(m_i, m_j)\big) = d + 1$.

Note that we will abuse notation and use M_d to mean both the vertices of $S_n(d)$ and the set of monomials of degree d in $R = k[x_1, \ldots, x_n]$.

Example 2.1. If we consider $S_3(3)$, then $x_1^3x_2$ and $x_1^3x_3$ are adjacent but $x_1^3x_2$ and $x_2^3x_3$ are not. The graph $S_3(3)$ is given in Figure 1.

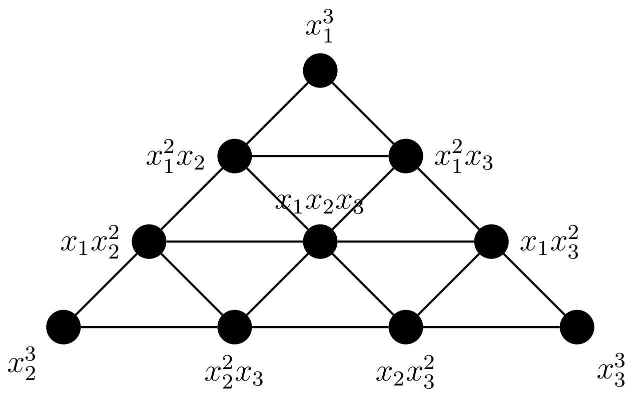

FIGURE 1. $S_3(3)$

The spreading and covering numbers are related to the maximum independent set and a special type of minimum clique covering, respectively, of $S_n(d)$.

Definition 2.2. A subset $V \subseteq M_d$ is an **independent set** if any two distinct elements of V are not adjacent; V is a **maximal independent set** if it is not properly contained in any larger independent set.

Definition 2.3. A subset of M_d in which any two vertices are adjacent is called a **clique**. A clique is **maximal** if it is not properly contained in any larger clique. If $C_1, \ldots, C_t$ are cliques, we say they form a **clique cover** of $S_n(d)$ if $C_1 \cup \cdots \cup C_t = M_d$. For any monomial m of degree $d-1$, an **upward clique** is the clique consisting of the vertices $mx_i \in M_d$ for all $x_i \in \{x_1, \ldots, x_n\}$.

As shown in [3], $\alpha_n(d)$ and $\rho_n(d+1)$ are equivalent to an invariant of $S_n(d)$:

Lemma 2.4. *With the notation as above*

(i) *$\alpha_n(d)$ is the cardinality of the largest maximal independent set of $S_n(d)$.*
(ii) *$\rho_n(d+1)$ is the minimum cardinality of an upward clique cover of the vertices of $S_n(d+1)$.*

Example 2.5. The graph $S_3(3)$ in Example 2.1 has $\alpha_3(3) = 4$ because $\{x_1^3, x_2^3, x_3^3, x_1x_2x_3\}$ forms a maximal independent set. For the same graph, we have $\rho_3(2+1) = 4$ because $C_1 = \{x_1^3, x_1^2x_2, x_1^2x_3\}$, $C_2 = \{x_2^3, x_1x_2^2, x_3x_2^2\}$, $C_3 = \{x_3^3, x_1x_3^2, x_2x_3^2\}$, and $C_4 = \{x_1^2x_2, x_1x_2^2, x_1x_2x_3\}$ form a minimal upward clique cover of $S_3(2+1)$.

Finding the size of the largest maximal independent set or a minimum clique cover of a graph are both NP-hard problems. This explains, in part, why computing $\alpha_n(d)$ and $\rho_n(d+1)$ is so difficult. However, the symmetry of $S_n(d)$ offers hope for using specialized techniques to improve our bounds on $\alpha_n(d)$ and $\rho_n(d+1)$.

Let $\mathrm{Sym}(n)$ denote the symmetric group on the set $\{1, 2, \ldots, n\}$. For any $\mathbf{x}^{\mathbf{a}} = x_1^{a_1} \cdots x_n{}^{a_n} \in M_d$ and $\sigma \in \mathrm{Sym}(n)$, let $\sigma(\mathbf{x}^{\mathbf{a}})$ denote the monomial obtained by permuting the indices $1, \ldots, n$ according to the permutation σ. This operation preserves many properties of sets of vertices; e.g., independent sets and clique covers are both unaffected.

Lemma 2.6. *If $W = \{m_{i_1}, \ldots, m_{i_s}\}$ is an independent set on $S_n(d)$, then the set*

$$\sigma(W) = \{\sigma(m_j) \mid m_j \in W\}$$

is also an independent set of $S_n(d)$ for any $\sigma \in \mathrm{Sym}(n)$.

Proof. Since W is an independent set, for any two $m_j, m_k \in W$, $\deg\big(\mathrm{lcm}(m_j, m_k)\big) \neq d+1$. Applying σ to each element in W results only in an exchange of indices; the exponents of each indeterminate, and thus the degree of the monomial, remain unchanged. Therefore, $\deg\big(\mathrm{lcm}(\sigma(m_j), \sigma(m_k))\big) \neq d+1$, so $\sigma(m_j)$ and $\sigma(m_k)$ are not adjacent. □

The proof that clique covers are preserved is similar. We will elaborate further on how the symmetry of $S_n(d)$ comes into play in Sections 3 and 4. For now we present a recursive lower bound for $\alpha_n(d)$ that uses another fact about the structure of $S_n(d)$.

Theorem 2.7. *For all $n \geq 2, d \geq 3$, $\alpha_n(d) \geq \alpha_n(d-2) + \alpha_{n-1}(d)$.*

Proof. Consider all the vertices of $S_n(d)$ divisible by x_1^2, i.e., those vertices which are labelled with a monomial of the form $x_1^{a_1} \cdots x_n^{a_n}$ with $a_1 \geq 2$. The induced graph on these vertices is isomorphic to $S_n(d-2)$. Also, consider all the vertices which are not divisible by x_1. The induced graph on these vertices is isomorphic to $S_{n-1}(d)$.

The graph $S_n(d)$ consists of these subgraphs, plus all the vertices whose corresponding monomials have the form $x_1x_2^{a_2} \cdots x_n^{a_n}$, i.e., the degree of x_1 is one. These extra monomials form a "buffer" between the two subgraphs, that is, no vertex in the subgraph $S_n(d-2)$ is adjacent to a vertex in the subgraph $S_{n-1}(d)$. Thus, the union of an independent set in $S_n(d-2)$ and an independent set in $S_{n-1}(d)$ will be an independent set in $S_n(d)$. From this observation we get $\alpha_n(d) \geq \alpha_n(d-2) + \alpha_{n-1}(d)$. □

We will compare various lower bounds for $\alpha_n(d)$ in Section 5. In general, the recursive lower bound is not as tight as the lower bound found in [3].

3. Upper Bound for the Spreading Number

We will describe how to find an upper bound on $\alpha_n(d)$ by using some techniques from commutative algebra. In particular, we first show that $\alpha_n(d)$ can be encoded as the dimension of particular ring and then describe how to bound this dimension.

We first take our graph theory problem of Lemma 2.4 and translate it into an new question. For any finite simple graph G with vertex set $V_G = \{z_1, \ldots, z_n\}$ and edge set E_G, we can associate to G a quadratic square-free monomial ideal called the **edge ideal**. In particular, given G, the edge ideal of G is then

$$I(G) = (z_iz_j \mid \{z_i, z_j\} \in E_G) \subseteq T = k[z_1, \ldots, z_n].$$

Understanding how the graph theory invariants of G are encoded into the invariants of $T/I(G)$ is an ongoing area of research (e.g., see [8]). In particular, it is known that

$$\dim T/I(G) = \alpha(G),$$

where $\alpha(G)$ is the **independence number** of G, that is, is the cardinality of the largest independent set. When $G = S_n(d)$, it follows by Lemma 2.4 that $\alpha(S_n(d)) = \alpha_n(d)$, the spreading number. We have thus proved:

Lemma 3.1. *Let $I(S_n(d)) = (z_{m_i}z_{m_j} \mid \{m_i, m_j\}$ is an edge of $S_n(d))$ in the ring $T = k[z_m \mid m \in M_d]$. Then*

$$\alpha_n(d) = \dim T/I(S_n(d)).$$

We therefore want to compute $\dim T/I(S_n(d))$. Computer algebra systems, such as `CoCoA` or *Macaulay2*, normally tackle this problem by computing the associated Hilbert-Poincaré series. This approach improves upon naive methods, but even for small cases, many computer algebra systems cannot compute this dimension.

Remark 3.2. Computing $\alpha_n(d)$ by computing $\dim T/I(S_n(d))$ was first described in [1]. Our initial hope was that improved computer hardware in the intervening ten years would allow us compute new values of $\alpha_n(d)$. This hope ended up being too optimistic. For example, computing $\alpha_5(5)$ in *Macaulay 2* requires us to compute the Hilbert-Poincaré series of an ideal in a ring with $\binom{5+5-1}{5} = 126$ variables.

Instead of using the Hilbert-Poincaré series approach to computing the dimension of a ring, we wish to make use of a system of parameters.

Definition 3.3. Let $I \subseteq R = k[x_1, \ldots, x_n]$ be a homogeneous ideal. A **partial homogeneous system of parameters (hsop)** of R/I is a finite sequence of homogeneous forms $F_1, \ldots, F_t$ with $\deg F_i > 0$ for each i such that

$$\dim(R/(I, F_1, \ldots, F_t)) = \dim R/I - t.$$

A partial homogeneous system of parameters is a **(complete) homogeneous system parameters** if $t = \dim R/I$.

If we could find a hsop for $T/I(S_n(d))$, then we would be able to compute $\dim T/I(S_n(d))$. However, the standard algorithms for computing a hsop (see [5]) require knowing $\dim R/I$ in order to compute its hsop. We can still bound $\alpha_n(d)$ using the following lemma.

Lemma 3.4. *Let $L_1, \ldots, L_t$ be any t linear forms of T. Then*

$$\dim T/(I(S_n(d)), L_1, \ldots, L_t) + t \geq \dim T/I(S_n(d)) = \alpha_n(d).$$

Moreover, we have equality if $L_1, \ldots, L_t$ forms a partial hsop.

Proof. The second part of the proof follows directly from the definition of a partial hsop. The first part of the proof follows from the more general fact that for any homogeneous ideal I in T and linear form $L \in T$, then $\dim T/(I, L) \geq \dim T/I - 1$. □

This lemma forms the basis of our algorithm to compute an upper bound on $\alpha_n(d)$. We want to judiciously pick a set of linear forms $\mathcal{L} = \{L_1, \ldots, L_t\}$ so that computing $\dim T/I(S_n(d), L_1, \ldots, L_t)$ is easier than computing $\dim T/I(S_n(d))$. At the same time, we want to pick our L_i's so that a large subset of $\mathcal{L}$ forms a partial hsop.

For the moment, let us suppose that we know how to pick $\mathcal{L}$; we can then present the pseudo-code for this approach:

Algorithm 3.5. Compute an upper bound for $\alpha_n(d)$
Input: n, d — The number of variables and degree of monomials, respectively.
Output: An upper bound for $\alpha_n(d)$

```
Step 1 Let $T = k[z_{m_1}, \ldots, z_{m_\ell}]$ where $M_d = \{m_1, \ldots, m_\ell\}$ is the set of $\binom{n+d-1}{d} = \ell$
       monomials of degree $d$, and let $I(S_n(d))$ be the edge ideal of $S_n(d)$ in $T$.
Step 2 Pick a suitable choice of linear forms $\mathcal{L} = \{L_1, \ldots, L_t\}$.
Step 3 Compute $\dim T/J$ where $J = I(S_n(d)) + (L \mid L \in \mathcal{L})$.
Step 4 Return $\dim T/J + |\mathcal{L}|$.
```

We still need to explain how to pick $\mathcal{L}$ in Step 2 of Algorithm 3.5. We describe three strategies and comment on their strengths and weakness.

3.1. **Random Linear Forms.**

Fix integers n and d, and assume that some oracle has given a crude lower bound on $\alpha_n(d)$ (e.g., [3] showed that $v_n(d)/n \leq \alpha_n(d)$ where $v_n(d) = \binom{n+d-1}{d} = |V(S_n(d))|$). Pick t to be equal to this lower bound. Form a $t \times v_n(d)$ matrix C with random entries taken from the field k, and let $\mathcal{L}$ be the following t linear equations:

$$C \begin{bmatrix} z_{m_1} \\ \vdots \\ z_{m_{v_n(d)}} \end{bmatrix}$$

We then make use of the following theorem of Kemper:

Theorem 3.6 ([5, Proposition 1])**.** *Suppose that $A = R/I$ has dimension n, and let $d_1, \ldots, d_t \in \mathbb{N}_{>0}$. Then the set*

$$\{(F_1, \ldots, F_t) : \dim(R/(I, F_1, \ldots, F_t)) = n - t\}$$

forms a Zariski open subset of $A_{d_1} \times \cdots \times A_{d_t}$.

Our linear forms $\mathcal{L}$, for a suitable choice of matrix C, belong to the Zariski open subset $A_1 \times \cdots \times A_1$. In other words, we expect most, if not all, of the elements of $\mathcal{L}$ to form a partial hsop. Indeed, our computer experiments on small values of n and

d appear to support this statement. However, the computation of $\dim T/J$ with $J = I(S_n(d)) + (L \mid L \in \mathcal{L})$ in Step 3 of our algorithm appears just as difficult as computing $\dim T/I(S_n(d))$. We had to abandon this approach because it was not practical.

3.2. **Using the symmetry.** The second construction of $\mathcal{L}$ makes use of the symmetry of the graph $S_n(d)$. Consider the cycle $\sigma = (12\cdots n-1) \in \mathrm{Sym}(n)$. For each vertex $m \in M_d$ of the graph $S_n(d)$, define the linear form:

$$L(m) = z_m + z_{\sigma(m)} + \cdots + z_{\sigma^{n-1}(m)} \in T.$$

Here, z_m is the variable of $T = k[z_m \mid m \in M_d]$ indexed by m. We make two observations:

Lemma 3.7. *Let $m_i, m_j \in M_d$. With the notation as above*

(i) $L(m_i) = L(m_j)$ if and only if $m_j = \sigma^t(m_i)$ for some $t \in \{0, \ldots, n-1\}$.

(ii) If $\deg(\gcd(m, \sigma(m), \ldots, \sigma^{n-1}(m))) = d-1$, then the class of $L(m)$ is a zero divisor on $T/I(S_n(d))$.

Proof. (i) Suppose that $L(m_i) = L(m_j)$. So

$$z_{m_i} + z_{\sigma(m_i)} + \cdots + z_{\sigma^{n-1}(m_i)} = z_{m_j} + z_{\sigma(m_j)} + \cdots + z_{\sigma^{n-1}(m_j)}.$$

So $z_{m_j} = z_{\sigma^t(m_i)}$ for some integer t, i.e., $m_j = \sigma^t(m_i)$. Conversely, if $m_j = \sigma^t(m_i)$, then for all $l \in \{0, \ldots, n-1\}$, $\sigma^l(m_j) = \sigma^l(\sigma^t(m_i)) = \sigma^{l+t}(m_i) = \sigma^p(m_i)$ were $p \equiv l+t \pmod{n-1}$ since σ has order $n-1$. The conclusion now follows.

(ii) We begin with the observation that if $m = x_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$, then

$$\sigma^i(m) = x_1^{a_{i+1}}x_2^{a_{i+2}}\cdots x_{n-i-1}^{a_{n-1}}x_{n-i}^{a_1}\cdots x_{n-1}^{a_i}x_n^{a_n}$$

Thus, if $a = \min\{a_1, \ldots, a_{n-1}\}$, then $\gcd(m_i, \sigma(m_i), \ldots, \sigma^{n-1}(m_i)) = x_1^a \cdots x_{n-1}^a x_n^{a_n}$. This monomial will only have degree $d-1$ if $n-2$ of $\{a_1, \ldots, a_{n-1}\}$ equal a, and the remaining exponent equals $a+1$. Say that $a_i = a+1$.

Consider the monomial $\tilde{m} = \frac{m}{x_i}x_n = x_1^a \cdots x_{n-1}^a x_n^{a_n+1}$. Let $z_{\tilde{m}}$ be the corresponding monomial in T. The linear forms $L(m)$ and $z_{\tilde{m}}$ are not in $I(S_n(d))$. However,

$$z_{\tilde{m}}L(m) = z_{\tilde{m}}(z_m + z_{\sigma(m)} + \cdots + z_{\sigma^{n-1}(m)}) \in I(S_n(d))$$

because $\tilde{m}$ and $\sigma^i(m)$ are adjacent in $S_n(d)$ for each $i = 0, \ldots, n-1$. This implies (ii). $\square$

Our second strategy for constructing $\mathcal{L}$ is to form the set

$$\mathcal{L} = \{L(m) \mid m \in M_d\} \setminus \{L(m) \mid \deg(\gcd(m, \sigma(m), \ldots, \sigma^{n-1}(m))) = d-1\}.$$

By Lemma 3.7 (i) we can pick all the $L(m)$ to be distinct. Moreover, we want to eliminate those linear forms that are zero divisors since they will never be part of partial hsop.

In contrast to our previous method, we found that the computation of $\dim T/J$ with $J = I(S_n(d)) + (L \mid L \in \mathcal{L})$ was significantly easier than the computation of $\dim T/I(S_n(d))$. It also required less memory. However, this method has the disadvantage of introducing too many extraneous linear forms, resulting in large upper bounds.

3.3. **Neighbours.** Our third choice for constructing $\mathcal{L}$ makes use of the neighbours of a vertex. Recall that for any $m \in M_d$, the **neighbourhood of** m, is the set $N(m) = \{m' \in M_d \mid \{m, m'\} \text{ is an edge of } S_n(d)\}$. To each $m \in M_d$ we associate the linear form

$$L(m) = z_m + \sum_{m' \in N(m)} z_{m'}.$$

Computer experimentation has then suggested the following construction of $\mathcal{L}$. Suppose that some oracle has given some maximal independent set W of vertices in the graph $S_n(d)$ (in the next section, we describe an algorithm which does this). Our desired set is

$$\mathcal{L} = \{L(m) : z_m \in W\}.$$

When we applied Algorithm 3.5 with this set of linear forms in Step 2, we found that for known values of n and d (i.e., small n and d), our bounds were close to the true value of $\alpha_n(d)$. In other words, $\mathcal{L}$ was almost a partial hsop. At the same time, the computation of $\dim T/J$ with $J = I(S_n(d)) + (L \mid L \in \mathcal{L})$ appeared easier than the computation of $\dim T/I(S_n(d))$. Unfortunately, as n and d increase, the computation of $\dim T/J$ bececomes difficult; see Section 5 for more details. As a final comment, it would be of interest to find a good theoretical reason why one should use a maximal independent set.

4. Greedy Algorithms for Computing Bounds

In this section, we use the symmetry of the graph $S_n(d)$ to either compute or place bounds on $\alpha_n(d)$ and $\rho_n(d)$. The symmetry is used to reduce the number of cases we must check in our algorithms. We first present a greedy lower bound algorithm for $\alpha_n(d)$. Following that, we will explain some additional ways to use the symmetry of $S_n(d)$ and present two greedy upper bound algorithms for $\rho_n(d+1)$.

4.1. **The Spreading Number.** Be Lemma 2.4 $\alpha_n(d)$ is the cardinality of the maximum independent set on $S_n(d)$. Finding a random maximal independent set is trivial with a greedy algorithm. Depending upon our initial choice of vertex and the graph itself, the difference between the cardinality of the resulting set and $\alpha_n(d)$ will vary greatly. The following algorithm is well known; we include it for completeness.

Algorithm 4.1. Compute a lower bound for $\alpha_n(d)$
Input: n, d — The number of variables and degree of monomials, respectively.
W — An initial independent set. (Optional)
Output: The cardinality of a maximal independent set on $S_n(d)$

```
Step 1 Let $L$ denote the set of all vertices not in $W$ and not adjacent to any $m \in W$.
Step 2 Select the vertex $v \in L$ with the smallest degree, and add it to $W$.
Step 3 Remove $v$ and $N(v)$ from $L$. If $L = \emptyset$, return $|W|$. Otherwise, return to Step 2.
```

We describe an alternative approach to finding a lower bound for $\alpha_n(d)$. The motivation comes from an observation about the case $n = 3$. Interior vertices of $S_3(d)$ are of degree 6, and if we regard the vertices of a maximum independent set as the centres of hexagons, then a maximum independent set on $S_3(d)$ yields a hexagonal tiling of the plane. Constructing this sort of tiling on $S_3(d)$ therefore yields a lower bound for $\alpha_3(d)$. In the next algorithm, we pick a vertex "in the middle" of $S_n(d)$. We then work out towards the exterior of $S_n(d)$; the vertices in our independent set can be thought of as the centre of n-dimensional tile.

Algorithm 4.2. Compute a lower bound for $\alpha_n(d)$

Input: n, d — The number of variables and degree of monomials, respectively.
Output: The cardinality of a maximal independent set on $S_n(d)$

```
Step 1 Let $W = \emptyset$ and $E = M_d$.
Step 2 Let $a = \lfloor \frac{d}{n} \rfloor$, and set $v_1 = x_1^a x_2^a \cdots x_{n-1}^a x_n^e$ with $e = d - (n-1)a$. Designate $v_1$ the
       **origin**. Add $v_1$ to $W$.
Step 3 Let $E := E \setminus (N(v_1) \cup \{v_1\})$ be the set of all vertices eligible for inclusion in
       $W$. Choose a vertex $v_2$ one degree removed from the origin, i.e, adjacent to some
       element of $N(v_1)$. Then $v_2$ is the new origin.
Step 4 Repeat the above step. At each step $i$, choose the new origin to be the vertex with
       the most neighbours already discarded from $E$. This preserves as many vertices as
       possible for consideration. If there are no eligible vertices one degree removed from
       the current origin, backtrack to the previous origin, and continue until $E = \emptyset$.
Step 5 Return $|W|$.
```

4.2. **The Covering Number.** By Lemma 2.4 $\rho_n(d+1)$ is the minimum upward clique cover of $S_n(d+1)$. As with finding maximal independent sets, we can use greedy algorithms to produce minimal clique covers of $S_n(d+1)$. These provide an upper bound for $\rho_n(d+1)$.

By definition, every upward clique is uniquely identified with a monomial from M_d. For vertices of $S_n(d+1)$ that consist of more than one indeterminate, many factorizations are possible; e.g., $x_1x_2^2$ can be written as $(x_1)x_2^2$ or $(x_1x_2)x_2$. For monomials of the form x_i^{d+1}, however, there is only one such factorization: $x_i^d x_i$. Thus, x_i^{d+1} belongs only to the upward clique identified with x_i^d. These unique upward cliques containing each x_i^{d+1} must therefore by in any clique cover of $S_n(d+1)$, so we can use them as the initial set when constructing an upward clique cover.

Algorithm 4.3. Compute an upper bound for $\rho_n(d)$.

Input: n, d — The number of variables and degree of monomials, respectively.
Output: A minimal upward clique cover of $S_n(d)$

```
Step 1 Initialize our cover $\mathcal{C}$ with the set of upward cliques that contain $x_i^d$.
Step 2 Select an uncovered vertex, $v$, at random. If there are no vertices left uncovered,
       go to Step 4.
Step 3 Iterate over the upward cliques that contain $v$. Select the upward clique with the
       fewest number of vertices already covered and add it to $\mathcal{C}$. Return to Step 2.
Step 4 For each $v \in M_d$, compute its frequency, i.e., the number of upward cliques that
       contain it, in $\mathcal{C}$.
Step 5 Iterate over the elements of the $\mathcal{C}$. If an upward clique does not contain a vertex
       of frequency 1—all its vertices are represented by other cliques as well—then it
```

is not essential to the cover, so discard it. Repeat this step until we complete an iteration without discarding any cliques.

Step 6 Return $\mathcal{C}$ as a minimal cover.

Aside from our choice of initial members of the cover, this method does not take into account any specific information about $S_n(d+1)$. Curtis [2] proved a formula for $\rho_3(d+1)$ using combinatorial methods, and [4] has a proof of the same result using linear programming. Hulett and Will also provided lower bounds for $\rho_4(d+1)$, when $d > 25$, and upper bounds for $\rho_4(d+1)$, when $d > 4$. They observed that, owing to the structure of $S_n(d)$, vertices on the interior of the graph can be covered exactly once, and the "excess" members of the cover come from vertices along the outside of the graph.

We adapted our greedy algorithm to this idea, removing the random selection of vertices to cover and instead working outward, covering the interior vertices first and the vertices closest to the outside last. We also use $\mathrm{Sym}(n)$ to create orbits of the vertices of $S_n(d)$.

Definition 4.4. For any $m \in M_d$, the **orbit of** m is the set

$$\{\sigma(m) \mid \sigma \in \mathrm{Sym}(n)\}.$$

Since elements of $\mathrm{Sym}(n)$ do not alter the exponents of a monomial, only the order of the exponents relative to the indeterminates, the orbit of m is also the set of all permutations of the exponents of m. By definition, the exponents of any $m \in M_d$ always sum to d, and therefore the orbits of $S_n(d)$ are in an one-to-one correspondence with the integer partitions of d of length at most n. We can write orbits as vectors in $\mathbb{N}^n$, and in this form it is easy to determine whether an orbit is an independent set, a clique, or neither by examining the entries in the vector. Iterating over the list of orbits of $S_n(d+1)$ in reverse lexicographical order will help us in computing an upper bound on $\rho_n(d+1)$.

Algorithm 4.5. Compute an upper bound for $\rho_n(d)$.

Input: n, d — The number of variables and degree of monomials, respectively.
Output: A minimal upward clique cover of $S_n(d)$

```
Step 1 Initialize our cover $\mathcal{C}$ with the set of upward cliques that contain $x_i^d$.
Step 2 Obtain a list, $L$, of the orbits of $S_n(d)$, where each orbit is represented as a vector
       in $\mathbb{N}^n$. Sort the list in reverse lexicographical order.
Step 3 Iterate over $L$. For each orbit $O \in L$, iterate over the vertices $v \in O$. If $v$ is
       covered, continue. If not, iterate over the upward cliques containing $v$. Select the
       upward clique that contains the fewest number of vertices already covered, and
       add it to $\mathcal{C}$.
Step 4 Minimize the cover by computing the frequency of each vertex and eliminating
       unnecessary upward cliques, as described in Algorithm 4.3.
Step 5 Return $\mathcal{C}$ as a minimal cover.
```

4.3. **From Bounds to Exact Values.** The greedy algorithms described for both $\alpha_n(d)$ and $\rho_n(d+1)$ can be extended, using a brute force approach, to compute the exact value of these numbers. For a set W of size $u \leq \alpha_n(d)$, we can iterate over the sets of size $u+1$ to check if a larger independent set exists. If we do not find such a set, then obviously $\alpha_n(d) = u$; otherwise, we can maximize the new independent set and repeat the process.

Similarly, for a cover $\mathcal{C}$ of size $t \geq \rho_n(d+1)$, we can iterate over families of cliques of size $t-1$ to check for a smaller cover. With each iteration, the number of combinations we are required to check grows quite quickly. We can reduce this number using the symmetry of the graph: each set $T \subseteq M_d$ is associated, via $\mathrm{Sym}(n)$, with $n!$ (not necessarily distinct) other subsets of M_d. Hence, when we eliminate T, we also eliminate up to $n!$ other potential sets that we must check. In practise, this improvement is not significant enough to make this method feasible. For example, the best lower bound we have for $\alpha_5(5)$ is 26. Since $|M_5| = 126$, we would have to check $\binom{126}{27} \approx 2.33 \times 10^{27}$ sets to ascertain whether $\alpha_n(d) = 26$. This is an improvement of several orders of magnitude upon the purely brute-force approach, where we would need to check 8.5×10^{37} sets. However, it is still computationally infeasible when run as a serial process, even on an HPC cluster provided by SHARCNET.

5. Results and Comparison to Known Bounds

In this section we will summarize the bounds for $\alpha_n(d)$ and $\rho_n(d+1)$ reported in previous papers and compare them to bounds we found with our algorithms from Sections 3 and 4. We conclude the section with two tables that present the known values of the spreading and covering numbers, as well as the best known bounds for certain n and d. Some of the computations were performed in *Macaulay2* 1.3.1 with 4 GB of memory allocated to 1 CPU and 1 node on SHARCNET's Saw cluster.[2] Other computations ran on the Kraken cluster[3] and used up to 16 GB of memory in *Macaulay2* 1.4. The run times are taken from *Macaulay2*'s `time` function.

5.1. **Existing Bounds.**

As well as providing the first definition of the spreading and covering numbers, Geramita, Gregory, and Roberts proved the following inequalities:

Theorem 5.1 ([3, Theorem 5.2 and Proposition 5.9]). *For all $n \geq 2, d \geq 2$, we have*

(1) $\frac{v_n(d)}{n} \leq \alpha_n(d) \leq \rho_n(d) \leq \frac{v_n(d)}{n} + \frac{n-1}{n} v_{n-1}(d)$, *and*
(2) $\frac{v_n(d)}{n-1} - \frac{\alpha_n(d-1)}{n-1} \leq \rho_n(d)$.

Moreover, they presented formulas for $\alpha_3(d)$ [3, Theorem 5.4], $\alpha_n(3)$ and $\rho_n(3)$ [3, Theorem 5.8], and $\alpha_4(d)$ for d odd [3, Proposition 5.6]. They include a formula, without proof, for the case of even d as well. We will refer to these formulas in Section 6 to prove the correspondence of $\alpha_4(d)$ to an interesting integer sequence. Curtis [2] proved a formula for $\rho_3(d+1)$, and in [4], Hulett and Will presented an alternative proof of this formula, as well as bounds on $\rho_4(d+1)$:

Theorem 5.2 ([4, Theorems 4.1 and 4.2]). *For all $d \geq 5$, we have*

(1) *if d is odd, $\rho_4(d) \leq (d^3 + 15d^2 - 61d + 261)/24$, or*
(2) *if d is even, $\rho_4(d) \leq (d^3 + 15d^2 - 34d + 240)/24$.*

[2] `https://www.sharcnet.ca/my/systems/show/41`
[3] `https://www.sharcnet.ca/my/systems/show/69`

5.2. **Comparison of Bounds for $\alpha_n(d)$.** We compared the lower bounds of [3] to our computations with Algorithm 4.2, as well as our random greedy algorithm. Like $\alpha_n(d)$ itself, the spreading number's existing upper bounds are similarly difficult to compute, so we focused on how closely Algorithm 3.5 approximates known values of $\alpha_n(d)$. In Tables 1 and 2, **GGR** refers to the lower bound for $\alpha_n(d)$ from in Theorem 5.1(1) , and **2.7** refers to the recursive lower bound from Theorem 2.7, while **4.1** and **4.2** refer to the bounds computed by those respective algorithms. Recall that Algorithm 4.1 has a random component, and so the reported result is the maximum taken over 100 iterations.

d	$\alpha_3(d)$	GGR	2.7	4.1	4.2
3	4	4	3	4	4
4	6	5	6	6	5
5	7	7	7	7	7
6	10	10	10	10	10
7	12	12	11	12	12
8	15	15	15	15	15
9	19	19	17	19	19
10	22	22	21	20	22

TABLE 1. Comparison of lower bounds for $\alpha_3(d)$.

d	$\alpha_4(d)$	GGR	2.7	4.1	4.2
3	5	5	5	5	5
4	11	9	10	11	9
5	14	14	12	14	13
6	24	21	21	24	19
7	30	30	26	27	27
8	45	42	39	40	35
9	55	55	49	48	48
10	76	72	67	64	60

TABLE 2. Comparison of lower bounds for $\alpha_4(d)$.

The GGR lower bound remains the best lower bound. For $n = 3$, the two algorithms perform quite well compared to this bound, while the bound of Theorem 2.7 is noticeably worse. For $n = 4$, the algorithms remain comparable with each other but are still worse than the GGR bound. Algorithm 4.2, which begins at the centre of the graph and works outward, produces worse lower bounds than the greedy Algorithm 4.1. Both algorithms are similar in terms of memory consumption and time complexity. As an example of the time needed, Algorithm 4.1 takes 0.40 seconds to compute a bound for $\alpha_4(3)$ but 40.99 seconds to compute a bound for $\alpha_4(10)$.

Overall, we have not made much progress improving lower bounds for $\alpha_n(d)$. Our algorithms are useful if one needs to generate a maximal independent set on $S_n(d)$, but if one needs only a bound, then the GGR formula remains superior.

Comparing our upper bounds from Section 3 with the bounds from [3] is more difficult. Geramita, Gregory, and Roberts prove several upper bounds that, like $\alpha_n(d)$, are linked to properties of $S_n(d)$ that make them difficult to compute. Our algorithms avoid this by using commutative algebra techniques; unfortunately, these methods introduce their own performance issues. However, our algorithms can compute an upper bound that, in tandem with a lower bound, provide a specific range into which each of $\alpha_n(d)$ must fall.

We tested two versions of Algorithm 3.5 differing in our choice of linear form. The first version uses the symmetry of $S_n(d)$, as described in Section 3.2. The second version uses neighbours of a vertex, as described in Section 3.3. We encountered practical limitations with both versions. The first version does not produce very tight bounds; for example, for $\alpha_4(8) = 40$, the algorithm gives 79 as an upper bound. The second version produces very good bounds, at least for the few values we could compute. This version consumes a great deal of memory. For $n = 3, 3 \leq d \leq 9$ this version gives bounds that match the known value of $\alpha_3(d)$. We ran out of memory computing a bound for $\alpha_3(10)$, and the same is true for $n = 4, d > 4$, and $n = 5, 6, d > 3$.

5.3. **Comparison of Bounds for $\rho_n(d+1)$.** We will examine how the upper bounds found in [3] and [4] compare to our computations with Algorithms 4.3 and 4.5. In Table 3, **GGR** refers to the upper bound for $\rho_n(d+1)$ from (1) in Theorem 5.1 and **HW** refers to the bounds from Theorem 5.2, while **4.3** and **4.5** refer to the bounds computed by those respective algorithms.

d	GGR	HW	4.3	4.5
5	30	19	22	19
6	42	33	33	29
7	57	38	44	40
8	75	60	62	55
9	97	69	81	74
10	121	100	113	96
11	150	114		122
12	182	155		147
13	219	175		185
14	260	227		223
15	306	254		275

TABLE 3. Comparison of upper bounds for $\rho_4(d)$.

All three revised bounds are better than the original GGR bound. Our original greedy algorithm, 4.3, fails to match HW. However, by taking advantage of the symmetry and structure of $S_n(d)$, our revised algorithm, Algorithm 4.5, is quite close to HW. In fact, it seems that for even d our bounds are equal or better, with the reverse is true for odd d. This pattern holds for at least $d \leq 24$, with the exception of $d = 22$. We are not certain why this is the case. Finally, recall that the HW bound holds only for $n = 4$, whereas our algorithm works for all $n \geq 2$. When tested against GGR for small values of d for $n = 5, 6$, Algorithm 4.5 consistently performs better (refer to Table 5 for example output).

Neither of our algorithms consume much memory, but as one might expect, as d increases the computational time increases significantly. As a result, we did not compute bounds for $d > 10$ with Algorithm 4.3: the computation for $d = 3$ took only 2.37 seconds, but for $d = 10$, the algorithm took 1709.44 seconds to terminate. Algorithm 4.5 once again performs better, taking 0.05 seconds for $d = 3$ and 21.28 seconds for $d = 10$. With each increase in degree, the amount of time to compute a bound roughly doubles, with $d = 24$ taking 83051.40 seconds. This complexity increases as n increases; for example, we computed a bound for $\rho_6(3)$ in 2.77 seconds, but a bound for $\rho_6(4)$ takes 7.81 seconds.

Our computations indicate that our algorithms yield tighter upper bounds for $\rho_n(d+1)$ than those of [3] and, $n = 4$ and d even, [4]. In some cases, it might be preferable to sacrifice some accuracy in favour of speed, particularly for values of n and d at which our algorithms might become infeasible. Even so, we hope this provides a useful example of how one can use the structure and symmetry of $S_n(d)$ along with a greedy algorithm to improve bounds on $\alpha_n(d)$ and $\rho_n(d+1)$.

5.4. **Tables of Results.** These tables are based on those from [1]. The $\alpha_4(d)$ row was filled according to the formulas in [3]. In some entries we have used a bracketed notation, $[a, b]$, to indicate a range given by known lower and upper bounds a and b, respectively. For values we could not compute exactly, we have attempted to bound them within a neighbourhood $[a, b]$ where $|b - a| \leq 100$. If no value is given, then we could not find such a neighbourhood.

d	1	2	3	4	5	6	7	8	9	10
$\alpha_1(d)$	1	1	1	1	1	1	1	1	1	1
$\alpha_2(d)$	1	2	2	3	3	4	4	5	5	6
$\alpha_3(d)$	1	3	4	6	7	10	12	15	19	22
$\alpha_4(d)$	1	4	5	11	14	24	30	45	55	76
$\alpha_5(d)$	1	5	7	16	[26,33]	[42,61]	[66,94]	[99,142]	[143,209]	[201,285]
$\alpha_6(d)$	1	6	10	[24,30]	[42,63]	[77,119]	[132,201]	[215,]	[334,]	[501,]
$\alpha_7(d)$	1	7	14							
$\alpha_8(d)$	1	8	[16,20]							
$\alpha_9(d)$	1	9								
$\alpha_{10}(d)$	1	10								

TABLE 4. Computed values and bounds of the spreading number

d	1	2	3	4	5	6	7	8	9	10
$\rho_1(d+1)$	1	1	1	1	1	1	1	1	1	1
$\rho_2(d+1)$	2	2	3	3	4	4	5	5	6	6
$\rho_3(d+1)$	3	4	6	9	12	15	18	23	27	32
$\rho_4(d+1)$	4	6	12	18	27	[32,38]	[45,55]	[58,69]	[77,96]	[96,114]
$\rho_5(d+1)$	5	9	20	33	[42,61]	[66,94]	[99,142]	[143,209]	[200,285]	[273,]
$\rho_6(d+1)$	6	12	30	[42,63]	[77,119]	[132,201]	[214,]	[333,]	[500,]	[728,]
$\rho_7(d+1)$	7	16	[30,46]	[66,102]	[132,210]					
$\rho_8(d+1)$	8	20								
$\rho_9(d+1)$	9	25								
$\rho_{10}(d+1)$	10	30								

TABLE 5. Computed values and bounds of the covering number

6. A053307

We conclude by showing a relationship between $\alpha_4(d)$ and a known integer sequence. For $\alpha_4(d)$, [3] gives an explicit formula; in particular

$$\alpha_4(d) = \begin{cases} \frac{v_4(d)}{4} & \text{for } d \text{ even} \\ \frac{v_4(d)}{4} + \frac{3d+6}{8} & \text{for } d \text{ odd.} \end{cases}$$

Here, $v_4(d)$ is the number of vertices of $S_4(d)$.

Theorem 6.1. *For $d \geq 0$, $\alpha_4(d)$ is the number of non-negative integer 2×2 matrices with sum of entries equal to d, under row and column permutations.*

Proof. Let $a(d)$ be the number of non-negative integer 2×2 matrices with sum of entries equal to d, under row and column permutations. This sequence is in the OEIS as A053307 [7]. In fact, it is an interleaved sequence where

$$a(2d+1) = \text{A000330}(d+1) = \frac{(d+1)(d+2)(2d+3)}{6}$$

and

$$a(2d) = \text{A006527}(d+1) = \frac{(d+1)^3 + 2(d+1)}{3}.$$

Recalling that $v_4(d) = \binom{d+3}{3}$, it follows that

$$\begin{aligned}\alpha_4(2d+1) &= \frac{v_4(2d+1)}{4} = \frac{\binom{2d+4}{3}}{4} = \frac{(2d+4)(2d+3)(2d+2)}{3!(4)} \\ &= \frac{(d+2)(2d+3)(d+1)}{6} = a(2d+1).\end{aligned}$$

Similarly,

$$\begin{aligned}\alpha_4(2d) &= \frac{v_4(2d)}{4} + \frac{3(2d)+6}{8} = \frac{\binom{2d+3}{3}}{4} + \frac{6d+6}{8} \\ &= \frac{8d^3+24d^2+22d+6}{24} + \frac{6d+6}{8} = \frac{8d^3+24d^2+40d+24}{24} \\ &= \frac{d^3+3d^2+5d+3}{3} = \frac{(d+1)^3+2(d+1)}{3} = a(2d).\end{aligned}$$

□

Even though the sequence A053307 and $\alpha_4(d)$ are related, it is not immediately apparent why this sequence and the spreading numbers of $S_4(d)$ are linked. The correspondence may be a result of the two interleaved sequences that make up A053307: the square pyramidal numbers, for odd d, and the sum of two tetrahedral numbers, for even d. Thus, we conclude with the following question.

Question 6.2. *What is the relationship between the spreading number in four variables and the sequence A053307? Why are they equivalent?*

Hopefully explaining the relationship between $\alpha_4(d)$ and A053007 could open up new techniques for computing the spreading and covering numbers.

References

[1] E. Carlini, H.T. Hà, A. Van Tuyl, Computing the spreading and covering numbers. Comm. Algebra **29** (2001) 5687–5699.

[2] F. Curtis, A combinatorial problem involving monomial ideals. J. Pure Appl. Algebra **104** (1995) 161–167.

[3] A. Geramita, D. Gregory, L. Roberts, Monomial Ideals and Points in Projective Space. J. Pure Appl. Algebra **40** (1986) 33–62.

[4] H. Hulett, T. Will, Generating monomials in dimensions three and four. J. Pure Appl. Algebra **138** (1999) 139–150.

[5] G. Kemper, An algorithm to calculate optimal homogeneous systems of parameters. J. Symbolic Comput. **27** (1999), 171–184.

[6] D. R. Grayson and M. E. Stillman, Macaulay 2, a software system for research in algebraic geometry. `http://www.math.uiuc.edu/Macaulay2/`.

[7] The On-Line Encyclopedia of Integer Sequences, `http://oeis.org/` (2010), sequences A000330, A006527, and A053307.

[8] R.H. Villarreal, *Monomial algebras.* Monographs and Textbooks in Pure and Applied Mathematics, **238**. Marcel Dekker, Inc., New York, 2001.

Department of Mathematical Sciences, Lakehead University, Thunder Bay, ON, P7B 5E1, Canada

E-mail address: `bababcoc@lakeheadu.ca`

Department of Mathematical Sciences, Lakehead University, Thunder Bay, ON, P7B 5E1, Canada

E-mail address: `avantuyl@lakeheadu.ca`