A Branch-and-Cut Strategy for the Manickam-Miklós-Singhi Conjecture

Stephen G. Hartke* Derrick Stolee†

February 14, 2013

Abstract

The Manickam-Miklós-Singhi Conjecture states that when $n \geq 4k$, every multiset of n real numbers with nonnegative total sum has at least $\binom{n-1}{k-1}$ k-subsets with nonnegative sum. We develop a branch-and-cut strategy using a linear programming formulation to show that verifying the conjecture for fixed values of k is a finite problem. To improve our search, we develop a zero-error randomized propagation algorithm. Using implementations of these algorithms, we verify a stronger form of the conjecture for all $k \leq 7$.

1 Introduction

Given a sequence $x_1, \ldots, x_n$ of real numbers with nonnegative sum $\sum_{i=1}^n x_i$, it is natural to ask how many partial sums must be nonnegative. Bier and Manickam [4] showed that every nonnegative sum $\sum_{i=1}^n x_i$ has at least 2^{n-1} nonnegative partial sums. Manickam, Miklós, and Singhi [10, 11] considered the situation when each partial sum has exactly k terms (we call such partial sums *k-sums*), and they conjectured there are many nonnegative k-sums when n is sufficiently large.

Conjecture 1 (Manickam-Miklós-Singhi [10, 11])**.** *Let $k \geq 2$ and $n \geq 4k$. For all $x_1, \ldots, x_n \in \mathbb{R}$ such that $\sum_{i=1}^n x_i \geq 0$, there are at least $\binom{n-1}{k-1}$ subsets $S \subset [n]$ with $|S| = k$ such that $\sum_{i \in S} x_i \geq 0$.*

The bound $\binom{n-1}{k-1}$ is sharp since the example $x_1 = n-1$ and $x_i = -1$ for $i \neq 1$ has sum zero and a k-sum is nonnegative exactly when it contains x_1. The bound $n \geq 4k$ is not necessarily sharp.

In this paper, we develop a computational method to solve fixed cases of the conjecture based on a poset of k-sums and a linear programming instance. In particular, we prove a

*Department of Mathematics, University of Nebraska–Lincoln, hartke@math.unl.edu. Supported by National Science Foundation Grant DMS-0914815.

†Department of Mathematics, University of Illinois, stolee@illinois.edu.

stronger statement than the conjecture for all $k \leq 7$. This leads us to formulate a stronger form of the conjecture (see Conjecture 2).

Conjecture 1 is trivial for $k = 1$ and is a simple exercise for $k = 2$. Marino and Chiaselotti [12] proved the conjecture for $k = 3$. Bier and Manickam [4] showed there exists a minimum integer $f(k)$ such that for all $n \geq f(k)$, there are at least $\binom{n-1}{k-1}$ nonnegative k-sums in any nonnegative sum of n terms. Subsequent results [3, 4, 5, 6, 15] give several exponential upper bounds on $f(k)$. Alon, Huang, and Sudakov [1] showed the polynomial bound $f(k) \leq \min\{33k^2, 2k^3\}$. Chowdhury [7] proved $f(3) = 11$ and $f(4) \leq 24$.

We prove $f(4) = 14$, $f(5) = 17$, $f(6) = 20$, and $f(7) = 23$. For $k \leq 7$ and $n < f(k)$, we find that the nonnegative sums with the fewest nonnegative k-sums have the following structure: for positive integers a and b summing to n, let $x_1 = \cdots = x_b = a$ and $x_{b+1} = \cdots = x_n = -b$. When $3k < n < f(k)$, the extremal examples have $a = 3$, $b = n - 3$, and $\binom{n-3}{k}$ nonnegative k-sums. This leads us to make the following conjecture.

Conjecture 2. *Let N_k be the smallest integer such that $\binom{N_k-3}{k} \geq \binom{N_k-1}{k-1}$. Then $f(k) = N_k$, and hence*[1]

$$\lim_{k\to\infty} \frac{f(k)}{k} = \lim_{k\to\infty} \frac{N_k}{k} \approx 3.147899.$$

In the next section we develop some necessary notation and preliminary results. In Section 2, we place a symmetry-breaking constraint on the vectors and investigate a natural poset among the k-sums. In Section 3, we develop our branch-and-cut strategy for verifying the conjecture. We then modify this algorithm by adding a randomized propagation algorithm in Section 4. In Section 5 we discuss stronger forms of the conjecture.

1.1 Preliminaries

Let $a_1, \ldots, a_\ell$ be real numbers and $e_1, \ldots, e_\ell$ be positive integers. We denote by $a_1^{e_1}\ a_2^{e_2}\ \ldots a_\ell^{e_\ell}$ the vector in $\mathbb{R}^n$ with the first e_1 coordinates equal to a_1, the next e_2 coordinates equal to a_2, and so on until the last e_ℓ coordinates are equal to a_ℓ. Using this notation, the vector $(n-1)^1\ (-1)^{n-1}$ is conjectured to achieve the minimum number of nonnegative k-sets when $n \geq 4k$.

For a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and a k-set S, define $\sigma_S(\mathbf{x}) = \sum_{i\in S} x_i$. The number of nonnegative k-sums of $\mathbf{x}$ is $s_k(\mathbf{x}) = \left|\left\{S \in \binom{[n]}{k} : \sigma_S(\mathbf{x}) \geq 0\right\}\right|$. For small values of k and n, we will compute the minimum integer $g(n,k)$ such that all vectors $\mathbf{x} \in \mathbb{R}^n$ with nonnegative sum have $s_k(\mathbf{x}) \geq g(n,k)$. By the sharpness example, $g(n,k) \leq \binom{n-1}{k-1}$ always. Define the *deficiency function* as $\hat{g}(n,k) = \binom{n-1}{k-1} - g(n,k)$.

Since removing the least coordinate from a vector $\mathbf{x} \in \mathbb{R}^n$ results in an $(n-1)$-coordinate vector with nonnegative sum, the function $g(n,k)$ is nondecreasing in n. Bier and Manickam [4] proved that $\hat{g}(n,k) = 0$ when k divides n by using a theorem of Baranyai [2].

[1] This limit was computed by first simplifying $\binom{x-3}{k} - \binom{x-1}{k-1} = 0$ to $(x-k)(x-k-1)(x-k-2) - k(x-1)(x-2) = 0$ and taking the real root for x of the resulting cubic.

The following lemma of Chowdhury [7] reduces the problem of computing $f(k)$ to verifying $\hat{g}(n,k) = 0$ for a finite number of values n.

Lemma 3 (Chowdhury [7])**.** *If $\hat{g}(n,k) = 0$, then $\hat{g}(n+k,k) = 0$.*

This lemma allows the computation of $f(k)$ by first computing a finite number of values of $g(n,k)$. We develop a finite process to verify $g(n,k) \geq t$ for fixed values of k and n, and hence by Lemma 3 we determine $f(k)$ when we find that $g(n,k) \geq \binom{n-1}{k-1}$ for k consecutive values of n. Our computation to verify $g(n,k) \geq t$ for fixed integers n, k, and t is designed as a search for a vector $\mathbf{x} \in \mathbb{R}^n$ with $s_k(\mathbf{x}) < t$. By constraining the order of the coordinates in a vector $\mathbf{x} \in \mathbb{R}^n$, we are able to make significant deductions about the sign of the k-sums of $\mathbf{x}$, which makes the computation very efficient. We first discuss this constraint on the coordinates of $\mathbf{x}$ in Section 2, and then discuss the algorithm in Section 3.

2 The Shift Poset

To better control the vectors $\mathbf{x} \in \mathbb{R}^n$ with nonnegative sum, we make a simple ordering constraint that adds significant structure. Let F_n be the set of vectors $\mathbf{x} \in \mathbb{R}^n$ with nonnegative sum and with the sequence x_i nondecreasing. That is,

$$F_n = \left\{ \mathbf{x} \in \mathbb{R}^n : \sum_{i=1}^{n} x_i \geq 0 \text{ and } x_1 \geq x_2 \geq \cdots \geq x_n \right\}.$$

Denote by $[n]$ the set $\{1, \ldots, n\}$ and by $\binom{[n]}{k}$ the collection of k-subsets of $[n]$. For two k-sets $S = \{i_1 < i_2 < \cdots < i_k\}$ and $T = \{j_1 < j_2 < \cdots < j_k\}$, let $S \succeq T$ if and only if $i_\ell \leq j_\ell$ for all $\ell \in \{1, \ldots, k\}$. We call this poset over $\binom{[n]}{k}$ the *shift poset*. Since we write $x_1, \ldots, x_n$ from left to right in nondecreasing order and associate a k-set S as with the values x_i for $i \in S$, then when $S \succeq T$ we say S is *to the left* of T and T is *to the right* of S. Observe that a relation $S \succeq T$ is a cover relation if and only if $S \setminus T = \{i\}$, $T \setminus S = \{j\}$, and $i = j - 1$.

Since we restrict vectors in F_n to have nondecreasing values as the index increases, $S \succeq T$ implies $\sigma_S(\mathbf{x}) \geq \sigma_T(\mathbf{x})$ for all $\mathbf{x} \in F_n$. Therefore, if $\sigma_T(\mathbf{x})$ is nonnegative, then so is $\sigma_S(\mathbf{x})$. Similarly, if $\sigma_S(\mathbf{x})$ is strictly negative, then so is $\sigma_T(\mathbf{x})$.

Given $S \in \binom{[n]}{k}$, let the *left shift* of S be $\mathcal{L}_k(S) = \{T \in \binom{[n]}{k} : T \succeq S\}$ and the *right shift* of S be $\mathcal{R}_k(S) = \{T \in \binom{[n]}{k} : S \succeq T\}$. Given a set $\mathcal{A} \subseteq \binom{[n]}{k}$, the *left closure* of $\mathcal{A}$, denoted $\mathcal{L}_k(\mathcal{A})$, is the union of all families $\mathcal{L}_k(S)$ over all sets $S \in \mathcal{A}$. Similarly, the *right closure* of $\mathcal{A}$, denoted $\mathcal{R}_k(\mathcal{A})$, is the union of all families $\mathcal{R}_k(S)$ over all sets $S \in \mathcal{A}$.

The shift poset has many interesting properties, including the fact that it is a lattice, which will be critical in later calculations. Fix any list $\mathcal{F} = \{S_1, \ldots, S_\ell\}$ of k-sets where $S_j = \{i_{j,1} < \cdots < i_{j,k}\}$. The *meet* of $\mathcal{F}$, denoted $\wedge\mathcal{F}$ or $\wedge_{j=1}^{\ell} S_j$, is the k-set T such that for all T' where $S_j \succeq T'$ for all $j \in [\ell]$, then $T \succeq T'$. The *join* of $\mathcal{F}$, denoted $\vee\mathcal{F}$ or $\wedge_{j=1}^{\ell} S_j$, is the k-set T such that for all T' where $T' \succeq S_j$ for all $j \in [\ell]$, then $T' \succeq T$. The meet and

join can be computed as

$$\wedge\mathcal{F} = \wedge_{j=1}^{\ell} S_j = \big\{ \max\{i_{j,t} : j \in [\ell]\} : t \in [k] \big\},$$
$$\vee\mathcal{F} = \vee_{j=1}^{\ell} S_j = \big\{ \min\{i_{j,t} : j \in [\ell]\} : t \in [k] \big\}.$$

Observe that the k-sets to the left of all $S_1, \ldots, S_\ell$ are exactly the k-sets to the left of the join $\vee_{j=1}^{\ell} S_j$, and the k-sets to the right of all $S_1, \ldots, S_\ell$ are exactly the k-sets to the right of the meet $\wedge_{j=1}^{\ell} S_j$. That is,

$$\bigcap_{j=1}^{\ell} \mathcal{L}_k(S_j) = \mathcal{L}\left(\vee_{j=1}^{\ell} S_j\right), \qquad \bigcap_{j=1}^{\ell} \mathcal{R}_k(S_j) = \mathcal{R}\left(\wedge_{j=1}^{\ell} S_j\right).$$

The shift poset preserves order with two standard linear orders of k-sets: the *lexicographic* and *colexicographic* orders. Let $\mathrm{lex}(S)$ and $\mathrm{colex}(S)$ be the *lexicographic* and *colexicographic* rank, respectively, of a k-set $S = \{i_1 < \cdots < i_k\} \subseteq \{1, \ldots, n\}$, defined as

$$\mathrm{lex}(S) = \sum_{\ell=1}^{k} \sum_{j=i_{\ell-1}+1}^{i_\ell - 1} \binom{n-j}{k-\ell}, \qquad \mathrm{colex}(S) = \sum_{\ell=1}^{k} \binom{i_\ell - 1}{\ell},$$

where $i_0 = 0$ by convention. These functions are bijections from $\binom{[n]}{k}$ to the set $\{0, \ldots, \binom{n}{k}-1\}$ with the property that when $S \succeq T$ we also have $\mathrm{lex}(S) \leq \mathrm{lex}(T)$ and $\mathrm{colex}(S) \leq \mathrm{colex}(T)$. In addition to these ranking functions, the lexicographic and colexicographic orders also have efficient *successor* and *predecessor* calculations. We use these methods to iterate over all k-sets to find $\succeq$-maximal or $\succeq$-minimal elements.

Finally, we note that the shift poset is isomorphic to the *dominance poset* over compositions of $n+1$ into $k+1$ positive parts. Given two compositions of $n+1$ into $k+1$ positive parts, where $A = \{a_1, \ldots, a_{k+1}\}$ and $B = \{b_1, \ldots, b_{k+1}\}$, $A \trianglelefteq B$ in the dominance order when $\sum_{\ell=1}^{t} a_\ell \leq \sum_{\ell=1}^{t} b_\ell$ for all $t \in [k]$. Note that the dominance order, also called the *majorization order*, is usually defined over partitions, but the defining inequalities are the same for compositions (see [14]).

For $S = \{i_1 < \cdots < i_k\}$, let $A_S = \{a_1, \ldots, a_{k+1}\}$ where $a_1 = i_1$, $a_\ell = i_\ell - i_{\ell-1}$ for $2 \leq \ell \leq k$, and $a_{k+1} = n + 1 - i_k$. The image A_S has the property that $\sum_{\ell=1}^{t} a_\ell = i_t$ for $t \leq k$ and $\sum_{\ell=1}^{k+1} a_\ell = n+1$. Therefore, if $S \succeq T$ in the shift poset, then $A_S \trianglelefteq A_T$ in the dominance order.

2.1 Counting Shifts

The functions $L_k(S) = |\mathcal{L}_k(S)|$ and $R_k(S) = |\mathcal{R}_k(S)|$ count the size of the left and right shifts, respectively. These shift functions are quickly computable using a recursive formula due to Chowdhury [7] which we prove for completeness. If $S = \{i_1, \ldots, i_k\}$ and $j \leq k$, let $S_j = \{i_1, \ldots, i_j\}$. We define $L_j(S) = |\mathcal{L}_j(S_j)|$.

Proposition 4 (Chowdhury [7]). *For a k-set $S = \{i_1 < i_2 < \cdots < i_k\} \in \binom{[n]}{k}$,*

$$L_k(S) = \binom{i_k}{k} - \binom{i_k - i_1}{k} - \sum_{\ell=1}^{k-2} L_\ell(S)\binom{i_k - i_{\ell+1}}{k-\ell}.$$

Proof. Observe that $\mathcal{L}_k(S) \subseteq \binom{[i_k]}{k}$. We will count the sets $T = \{j_1 < \cdots < j_k\}$ in $\binom{[i_k]}{k} \setminus \mathcal{L}_k(S)$ as $\binom{i_k - i_1}{k} + \sum_{\ell=1}^{k-2} L_\ell(S_\ell)\binom{i_k - i_{\ell+1}}{k-\ell}$. Every such set T has some minimum parameter $\ell \geq 0$ such that $j_{\ell+1} > i_{\ell+1}$. If $\ell = 0$, then the minimum of T is strictly larger than i_1 and there are exactly $\binom{i_k - i_1}{k}$ such sets. When $\ell \geq 1$, we have $j_t \leq i_t$ for all $t \leq \ell$. These sets can be constructed from the $L_\ell(S_\ell)$ ℓ-subsets of $[n]$ that are to the left of S_ℓ and extended by exactly $\binom{i_k - i_{\ell+1}}{k-\ell}$ $(k-\ell)$-subsets of $\{i_{\ell+1}+1, \ldots, i_k\}$. ∎

Observe that if $S = \{i_1 < \cdots < i_k\}$, then by symmetry for $T = \{n + 1 - i_\ell : \ell \in [k]\}$ we have $R_k(S) = L_k(T)$. While computing $L_k(S)$ for all k-sets, we first compute and store the values of $L_j(S_j)$ for $1 \leq j < k$.

2.2 Required Negative Sets

We use the shift function $L_k(S)$ to determine that certain k-sums must be negative in order to avoid t nonnegative k-sums.

Observation 5. *Let n, k, t be integers. If $S \in \binom{[n]}{k}$ has $L_k(S) \geq t$, then every vector $\mathbf{x} \in F_n$ with $s_k(\mathbf{x}) < t$ has $\sigma_S(\mathbf{x}) < 0$.*

Lemma 6. *Fix integers n, k, t with $t \leq \binom{n-1}{k-1}$ and a k-set $S \in \binom{[n]}{k}$ with $1 \in S$. If $L_k(S) + g(n-k, k) \geq t$, then every vector $\mathbf{x} \in F_n$ with $s_k(\mathbf{x}) < t$ has $\sigma_S(\mathbf{x}) < 0$.*

Proof. Fix a vector $\mathbf{x} \in F_n$ with $s_k(\mathbf{x}) < t$. The k-set $T = \{1, n-k+2, \ldots, n\}$ has $L_k(T) = \binom{n-1}{k-1}$, so by Observation 5, $\sigma_T(\mathbf{x}) < 0$. Since $\sum_{i=1}^{n} x_i \geq 0$, we have $\sum_{i=2}^{n-k+1} x_i = \sum_{i \notin T} x_i \geq 0$. Let $\mathcal{A}$ be the family of nonnegative k-sums over $x_2, \ldots, x_{n-k}$. By the definition of g, $|\mathcal{A}| \geq g(n-k, k)$. For all $Q \in \mathcal{A}$, the minimum element of Q is at least 2 and hence Q is not to the left of S, since the minimum element of S is 1. Therefore, if $\sigma_S(\mathbf{x}) \geq 0$, then $s_k(\mathbf{x}) \geq |\mathcal{L}_k(S)| + |\mathcal{A}| \geq L_k(S) + g(n-k, k) \geq t$, a contradiction. ∎

Similar inferences, discussed in Section 3.3, will be important to the performance of our algorithms.

3 The Branch-and-Cut Method

Our method to verify the conjecture centers on searching for vectors $\mathbf{x} \in F_n$ which are counterexamples (i.e. $s_k(\mathbf{x}) < \binom{n-1}{k-1}$). In addition, for values of n less than $f(k)$, we will search for vectors with the fewest number of nonnegative k-sets. For integers n, k, and t, we say a vector $\mathbf{x} \in F_n$ is (n, k, t)-*bad* if $s_k(\mathbf{x}) < t$. Using the shift poset, we will determine properties of such an (n, k, t)-bad vector and use that to guide our search.

For a vector $\mathbf{x} \in F_n$, the k-subsets of $[n]$ are partitioned into two parts by whether the associated k-sums are nonnegative or strictly negative. Let $\mathcal{C}_{\mathbf{x}}^+$ contain the k-sets S with $\sigma_S(\mathbf{x}) \geq 0$ and $\mathcal{C}_{\mathbf{x}}^-$ contain the the k-sets T with $\sigma_T(\mathbf{x}) < 0$. Observe $\mathcal{C}_{\mathbf{x}}^+ = \mathcal{L}_k(\mathcal{C}_{\mathbf{x}}^+)$ and $\mathcal{C}_{\mathbf{x}}^- = \mathcal{R}_k(\mathcal{C}_{\mathbf{x}}^-)$. Our computational method focuses on testing whether vectors $\mathbf{x} \in F_n$ exist, given certain contraints on $\mathcal{C}_{\mathbf{x}}^+$ and $\mathcal{C}_{\mathbf{x}}^-$.

Definition 7. Given families $\mathcal{A}^+$ and $\mathcal{A}^-$ of k-sets, let the linear program $P(k, n, \mathcal{A}^+, \mathcal{A}^-)$ be defined as

$$
\begin{aligned}
P(k, n, \mathcal{A}^+, \mathcal{A}^-): \qquad \text{minimize} \quad & x_1 \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i \geq 0 \\
& x_i - x_{i+1} \geq 0 \qquad \forall i \in \{1, \ldots, n-1\} \\
& \sum_{i \in S} x_i \geq 0 \qquad \forall S \in \mathcal{A}^+ \\
& \sum_{i \in T} x_i \leq -1 \quad \forall T \in \mathcal{A}^- \\
& x_1, \ldots, x_n \in \mathbb{R}
\end{aligned}
$$

A vector $\mathbf{x} \in F_n$ has $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{L}_k(\mathcal{A}^+)$ and $\mathcal{C}_{\mathbf{x}}^- \supseteq \mathcal{R}_k(\mathcal{A}^-)$ if and only if an appropriate scalar multiple of $\mathbf{x}$ is a feasible solution to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$.

Due to the following observations, we can verify that $s_k(\mathbf{x}) \geq t$ for the infinite set of vectors $\mathbf{x}$ in F_n by searching for partitions of $\binom{[n]}{k}$ that correspond to the nonnegative and negative k-sums of $\mathbf{x}$.

Observation 8. *If $\mathcal{L}_k(\mathcal{A}^+) \cup \mathcal{R}_k(\mathcal{A}^-) = \binom{[n]}{k}$, then every feasible solution $\mathbf{x}$ to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ has $s_k(\mathbf{x}) = |\mathcal{L}_k(\mathcal{A}^+)|$. In particular, a feasible solution to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ is (n, k, t)-bad if and only if $|\mathcal{L}_k(\mathcal{A}^+)| < t$.*

Observation 9. *If $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ is infeasible, then no vector $\mathbf{x} \in F_n$ has $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{L}_k(\mathcal{A}^+)$ and $\mathcal{C}_{\mathbf{x}}^- \supseteq \mathcal{R}_k(\mathcal{A}^-)$.*

Observation 10. *If $|\mathcal{L}_k(\mathcal{A}^+)| \geq t$, then $s_k(\mathbf{x}) \geq t$ for all feasible solutions to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$.*

By searching for sets $\mathcal{A}^+, \mathcal{A}^-$ such that a solution to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ has $s_k(\mathbf{x}) < t$, we either find these vectors or determine that none exist. Thus, determining $g(n, k)$ is a finite problem, and by Lemma 3 determining $f(k)$ is a finite problem.

3.1 The Search Strategy

Fix n, k, and $t \leq \binom{n-1}{k-1}$. We will search for sets $\mathcal{A}^+$ and $\mathcal{A}^-$ such that the solution $\mathbf{x}$ to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ has $s_k(\mathbf{x}) < t$. If $\mathcal{L}_k(\mathcal{A}^+) \cup \mathcal{R}_k(\mathcal{A}^-)$ is a partition of $\binom{[n]}{k}$ with $|\mathcal{L}_k(\mathcal{A}^+)| < t$, then every feasible solution to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ is (n, k, t)-bad. The reason we use the

objective of minimizing x_1 in the linear program is to try and distance the optimal solution from the conjectured sharp example of x_1 being a large positive value and $x_2, \ldots, x_n$ being small negative values. In practice, when an (n, k, t)-bad vector exists we discover it before $\mathcal{L}_k(\mathcal{A}^+)$ and $\mathcal{R}_k(\mathcal{A}^-)$ partition $\binom{[n]}{k}$.

During the algorithm, we will store two collections of k-sets, $\mathcal{B}^+$ and $\mathcal{B}^-$, called *branch sets*. We initialize $\mathcal{B}^+ = \mathcal{B}^- = \varnothing$ and always $\mathcal{B}^+ \subseteq \mathcal{A}^+$ and $\mathcal{B}^- \subseteq \mathcal{A}^-$. The difference between $\mathcal{B}^-$ and $\mathcal{A}^-$, for instance, is that the sets in $\mathcal{B}^-$ are *chosen* to have negative sum, but the sets in $\mathcal{A}^- \setminus \mathcal{B}^-$ are sets that *must* have negative sum for all (n, k, t)-bad vectors $\mathbf{x}$ with $\mathcal{C}^+_{\mathbf{x}} \supseteq \mathcal{B}^+$ and $\mathcal{C}^-_{\mathbf{x}} \supseteq \mathcal{B}^-$.

The search procedure BranchAndCut (Algorithm 1) is recursive, taking parameters $n, k, t, \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}^*$ under the requirement that $\mathcal{L}_k(\mathcal{B}^+) \cup \mathcal{R}_k(\mathcal{B}^-) \cup \mathcal{C}^*$ is a partition of $\binom{[n]}{k}$. The collection $\mathcal{C}^* = \binom{[n]}{k} \setminus (\mathcal{L}_k(\mathcal{B}^+) \cup \mathcal{R}_k(\mathcal{B}^-))$ contains the k-sets S where $\sigma_S(\mathbf{x})$ is not immediately decided by the constraints on $\mathcal{B}^+$ and $\mathcal{B}^-$.

We will refer to each recursive call to BranchAndCut as a *search node*, and at each search node we perform the following three actions:

1. Determine the $\succeq$-maximal sets $S \in \mathcal{C}^*$ such that all vectors $\mathbf{x}$ with $\mathcal{C}^+_{\mathbf{x}} \supseteq \mathcal{B}^+ \cup \{S\}$ have $s_k(\mathbf{x}) \geq t$ and add these sets to $\mathcal{A}^-$.

2. Test that the linear program $P(n, k, \mathcal{B}^+, \mathcal{A}^-)$ is feasible (prune if infeasible).

3. Select a set $S \in \mathcal{C}^*$ and branch on the choice of placing S in $\mathcal{B}^+$ or $\mathcal{B}^-$, creating two new search nodes.

One crucial step for the first action performed at each node is computing the number of k-sets in $\mathcal{C}^*$ that are also to the left of a set S or to the right of a set S. Define the functions $L^*_k(S) = |\mathcal{L}_k(S) \setminus \mathcal{L}_k(\mathcal{A}^+)|$ and $R^*_k(S) = |\mathcal{R}_k(S) \setminus \mathcal{R}_k(\mathcal{A}^-)|$.

The algorithm SelectBranchSet is a method for selecting a set $S \in \mathcal{C}^*$ for the branching step is called a *branching rule*. A good branching rule can significantly reduce the number of visited search nodes. However, it is difficult to predict which set is the best choice. Any branching rule that selects a k-set S from $\mathcal{C}^*$ will provide a *correct* algorithm, but these choices can greatly change the size and structure of the search tree. Based on our experiments and choices of several branching rules, we found selecting a k-set S that maximizes $\min\{L^*_k(S), R^*_k(S)\}$ was most effective. This rule ensured that both branches removed as many sets from $\mathcal{C}^*$ as possible.

In Section 3.3, we define the algorithm PropagateNegative (Algorithm 2). Before that, we must discuss how to efficiently compute $L^*_k(S)$ and $R^*_k(S)$.

3.2 Computing Intersections with $\mathcal{C}^*$

For this discussion, fix two families $\mathcal{A}^+$ and $\mathcal{A}^-$ with $\mathcal{C}^* = \binom{[n]}{k} \setminus (\mathcal{L}_k(\mathcal{A}^+) \cup \mathcal{R}_k(\mathcal{A}^-))$. We will use three methods to compute $|\mathcal{L}_k(S) \cap \mathcal{C}^*|$ and $|\mathcal{R}_k(S) \cap \mathcal{C}^*|$: breadth-first-search, inclusion-exclusion, and iterative updates. In all techniques, we will store markers for whether a set T is in $\mathcal{L}_k(\mathcal{A}^+)$, $\mathcal{R}_k(\mathcal{A}^-)$, or $\mathcal{C}^*$.

Algorithm 1 The recursive algorithm BranchAndCut$(n, k, t, \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}^*)$.

Require: $\mathcal{L}_k(\mathcal{B}^+)$, $\mathcal{R}_k(\mathcal{B}^-)$, and $\mathcal{C}^*$ partition $\binom{[n]}{k}$.
Ensure: Returns an (n, k, t)-bad vector $\mathbf{x}$ with $\mathcal{C}^+_{\mathbf{x}} \supseteq \mathcal{B}^+$ and $\mathcal{C}^-_{\mathbf{x}} \supseteq \mathcal{B}^-$ if it exists.
  if $|\mathcal{L}(\mathcal{B}^+)| \geq t$ **then**
    return Null
  end if
  $\mathcal{A}^+, \mathcal{A}^-, \mathcal{C}^* \leftarrow$ PropagateNegative$(n, k, t, \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}^*)$ *(See Algorithm 2)*
  Find optimal solution $\mathbf{x}$ to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$
  if $\mathbf{x} \equiv$ Null **or** $s_k(\mathbf{x}) < t$ **then**
    The linear program is infeasible or found a counterexample.
    return $\mathbf{x}$
  end if
  $S \leftarrow$ SelectBranchSet$(\mathcal{A}^+, \mathcal{A}^-, \mathcal{C}^*)$
  $\mathcal{B}^+_1 \leftarrow \mathcal{B}^+, \quad \mathcal{B}^-_1 \leftarrow \mathcal{B}^- \cup \{S\}, \quad \mathcal{C}^*_1 \leftarrow \mathcal{C}^*_0 \setminus \mathcal{R}_k(S)$
  $\mathbf{x} \leftarrow$ BranchAndCut$(n, k, t, \mathcal{B}^+_1, \mathcal{B}^-_1, \mathcal{C}^*_1)$
  if $\mathbf{x} \equiv$ Null **then**
    $\mathcal{B}^+_2 \leftarrow \mathcal{B}^+ \cup \{S\}, \quad \mathcal{B}^-_2 \leftarrow \mathcal{B}^-, \quad \mathcal{C}^*_2 \leftarrow \mathcal{C}^* \setminus \mathcal{L}_k(S)$
    $\mathbf{x} \leftarrow$ BranchAndCut$(n, k, t, \mathcal{B}^+_2, \mathcal{B}^-_2, \mathcal{C}^*_2)$
  end if
  return $\mathbf{x}$

Breadth-First-Search. When k is small, it is reasonable to store the Hasse digraph (an edge $S \leftarrow T$ exists if $S \succeq T$ is a cover relation) of the shift poset in memory and use breadth-first-search to count the number of sets to the left of S which are in $\mathcal{C}^*$.

Inclusion-Exclusion. We can use the lattice structure of the shift poset to compute the functions $L^*_k(S)$ and $R^*_k(S)$. First, we compute the values of $L^*_k(S)$ with increasing colex rank and values of $R^*_k(S)$ with decreasing colex rank. With this order, we have access to $L^*_k(T)$ whenever $T \succeq S$ or $R^*_k(T)$ whenever $S \succeq T$. Let $\mathcal{F}_L(S)$ be the collection of sets T that cover S, and let $\mathcal{F}_R(S)$ be the collection of sets T that S covers. Then,

$$L^*_k(S) = 1 + \sum_{\emptyset \neq \mathcal{A} \subseteq \mathcal{F}_L(S)} (-1)^{|\mathcal{A}|+1} L^*_k(\vee \mathcal{A})$$

$$R^*_k(S) = 1 + \sum_{\emptyset \neq \mathcal{A} \subseteq \mathcal{F}_R(S)} (-1)^{|\mathcal{A}|+1} R^*_k(\wedge \mathcal{A})$$

When $k \leq 4$, the breadth-first-search strategy is faster than the inclusion-exclusion strategy. However, for $k \geq 6$, the inclusion-exclusion strategy is the only method that is tractable. For $k = 5$, the two strategies are too similar to demonstrate a clear preference.

As k grows, the number of terms of the inclusion-exclusion sum grows exponentially, leading to a large amount of computation required for every set $S \in \mathcal{C}^*$. Our next method computes the values of $L^*_k(S)$ when a small change has been made to $\mathcal{B}^+$ using a smaller amount of computation for every set $S \in \mathcal{C}^*$.

Updates with $\mathcal{B}^+$. Observe that sets are added to $\mathcal{A}^+$ only when placing a set into $\mathcal{B}^+$. Therefore, there are many fewer sets in $\mathcal{A}^+$ than in $\mathcal{A}^-$. Further, if the most recent branch selected a k-sum to be negative, then the values $L_k^*(S)$ did not change for any sets still in $\mathcal{C}^*$. Therefore, we need only update the values of $L_k^*(T)$ for $T \in \mathcal{C}^*$ after branching on a set S and placing S in $\mathcal{B}^+$. For all such T, observe that

$$L_k^*(T) \leftarrow L_k^*(T) - L_k^*(T \vee S)$$

properly updates the value of $L_k^*(T)$ when adding the set S to $\mathcal{B}^+$. Specifically, the sets that are removed from $\mathcal{L}_k(T) \setminus \mathcal{L}(\mathcal{A}^+)$ to $\mathcal{L}_k(T) \setminus \mathcal{L}_k(\mathcal{A}^+ \cup \{S\})$ are exactly those to the left of both T and S but not to the left of any elements in $\mathcal{A}^+$. Such k-sets are exactly those in $\mathcal{L}_k(T \vee S) \setminus \mathcal{L}(\mathcal{A}^+)$, which are counted by $L_k^*(T \vee S)$.

This method is not efficient for computing $R_k^*(S)$, since too many sets are being added to $\mathcal{A}^-$ during the propagation step (see the next section). Thus, we use this method to update $L_k^*(S)$ only when a single set has been added to $\mathcal{B}^+$. If we need to recompute all values of $L_k^*(S)$ after a significant change to $\mathcal{B}^+$, we use the inclusion-exclusion method.

3.3 Propagation

Given a set of branch sets $\mathcal{B}^+$ and $\mathcal{B}^-$, we want to determine which sets $S \in \mathcal{C}^*$ must have $\sigma_S(\mathbf{x}) < 0$ for all (n,k,t)-bad vectors $\mathbf{x}$ which are feasible in $P(n,k,\mathcal{A}^+,\mathcal{A}^-)$.

Recall that by Observation 5 and Lemma 6, any set S with $L_k(S) \geq t$, or $1 \in S$ and $L_k(S) + g(n-k,k) \geq t$, has $\sigma_S(\mathbf{x}) < 0$ for any (n,k,t)-bad vector $\mathbf{x}$. This applies regardless of the previous choices of sets placed in $\mathcal{B}^+$. When $\mathcal{B}^+$ is non-empty and we have access to the function $L_k^*(S)$, we may be able to find more sets where $\sigma_S(\mathbf{x}) < 0$ for any (n,k,t)-bad vector $\mathbf{x}$ with $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{B}^+$.

Lemma 11. *If S is a k-set with $L_k^*(S) + |\mathcal{L}_k(\mathcal{B}^+)| \geq t$, then all feasible solutions $\mathbf{x}$ to $P(n,k,\mathcal{B}^+ \cup \{S\},\mathcal{B}^-)$ have $s_k(\mathbf{x}) \geq t$.*

Proof. Observe that such vectors $\mathbf{x}$ have $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{L}_k(\mathcal{B}^+) \cup \mathcal{L}_k(S)$ and $|\mathcal{L}_k(\mathcal{B}^+) \cup \mathcal{L}_k(S)| = |\mathcal{L}_k(\mathcal{B}^+)| + |\mathcal{L}_k(S) \setminus \mathcal{L}_k(\mathcal{B}^+)| = |\mathcal{L}_k(\mathcal{B}^+)| + L_k^*(S) \geq t$. ∎

The PropagateNegative algorithm (Algorithm 2) iterates on all sets S in $\mathcal{C}^*$ using lexicographic order and whenever a set S satisfies the hypotheses of Lemma 6 or Lemma 11 the set S is added to $\mathcal{A}^-$. Since all sets $T \succeq S$ have $T \leq_{\text{lex}} S$, the set T was considered before S and did not satisfy the condition for addition to $\mathcal{A}^-$. Hence, we construct $\mathcal{A}^-$ as a $\succeq$-maximal set (other than possible comparisons within $\mathcal{B}^-$).

3.4 Results Using BranchAndCut

Using BranchAndCut (Algorithm 1), we computed $g(n,k)$ for $k \in \{4,5,6\}$ and $k < n \leq 4k+1$, and we verified that $f(k) = 3k+2$ for $k \in \{4,5,6\}$.

Theorem 12. *$f(4) = 14$, $f(5) = 17$, and $f(6) = 20$.*

Algorithm 2 PROPAGATENEGATIVE$(n, k, t, \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}^*)$

Require: $\mathcal{L}_k(\mathcal{B}^+), \mathcal{R}_k(\mathcal{B}^-)$, and $\mathcal{C}^*$ partition $\binom{[n]}{k}$ and $|\mathcal{L}_k(\mathcal{B}^+)| < t$.
Ensure: Returns $\mathcal{B}^+$, $\mathcal{A}_0^-$, and $\mathcal{C}_0^*$ such that any (n, k, t)-bad vector $\mathbf{x}$ with $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{B}^+$ and $\mathcal{C}_{\mathbf{x}}^- \supseteq \mathcal{B}^-$ also has $\mathcal{C}_{\mathbf{x}}^- \supseteq \mathcal{A}_0^-$, and $\mathcal{C}_0^* = \binom{[n]}{k} \setminus (\mathcal{L}_k(\mathcal{B}^+) \cup \mathcal{R}_k(\mathcal{A}_0^-))$.

```
A_0^- ← B^-,   C_0^* ← C^*
for all sets S ∈ C_0^* (in lex order) do
  if 1 ∈ S and L_k(S) + g(n − k, k) ≥ t then
    A_0^- ← A_0^- ∪ {S}
    C_0^* ← C_0^* \ R_k(S)
  end if
  if L_k^*(S) + |L_k(B^+)| ≥ t then
    A_0^- ← A_0^- ∪ {S}
    C_0^* ← C_0^* \ R_k(S)
  end if
end for
return B^+, A_0^-, C_0^*
```

Proof outline. By Lemma 3, we must only verify the values n where $f(k) \le n \le f(k)+k-1$. For all $n \le 3k+1$, we tested all sums $a+b=n$ and found $s_k(\mathbf{x})$ for the vector $x_1 = \cdots = x_b = a$, $x_{b+1} = \cdots = x_n = -b$. We let t be the minimum such value $s_k(\mathbf{x})$ and executed BRANCHANDCUT$(n, k, t, \varnothing, \varnothing, \binom{[n]}{k})$, which found no vector with fewer than t nonnegative k-sums. By executing these algorithms in increasing value of n, we have access to $g(n-k, k)$ during execution of PROPAGATENEGATIVE$(n, k, t, \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}^*)$ (Algorithm 2) when testing $g(n, k) \ge t$ for $n > 2k$. Finally, for $n \in \{3k+2, \ldots, 4k+1\}$, we find $g(n, k) \ge \binom{n-1}{k-1}$ by executing BRANCHANDCUT$(n, k, \binom{n-1}{k-1}, \varnothing, \varnothing, \binom{[n]}{k})$. ∎

For $k = 7$, BRANCHANDCUT succeeded in computing $g(n, k)$ for $n \le 23$, but only after resorting to parallel computation. Previous computations showed an order of magnitude jump in computation time between $n = f(k) - 1$ and $n = f(k)$, so using BRANCHANDCUT to verity $\hat{g}(24, 7) = 0$ seemed intractable.

In the next section, we develop a zero-error randomized propagation algorithm which improves the performance enough for us to compute $f(7)$.

4 Propagation With Randomness

When branching in our branch-and-cut method, we select a set S in $\mathcal{C}^*$ and make a temporary decision of $\sigma_S(\mathbf{x}) \le -1$ or $\sigma_S(\mathbf{x}) \ge 0$. Then during our propagation step, the PROPAGATENEGATIVE algorithm (Algorithm 2) places as many left-most sets into $\mathcal{A}^-$ as possible using Lemmas 6 and 11. We now describe a condition that allows us to add sets into $\mathcal{A}^+$ without branching.

Algorithm 3 PROPAGATEPOSITIVE$(n, k, t, \mathcal{A}^+, \mathcal{A}^-, \mathcal{C}^*)$

Require: $\mathcal{L}_k(\mathcal{A}^+), \mathcal{R}_k(\mathcal{A}^-)$, and $\mathcal{C}^*$ partition $\binom{[n]}{k}$ and $|\mathcal{L}_k(\mathcal{A}^+)| < t$.
Ensure: Returns $\mathcal{A}_0^+$, $\mathcal{A}_0^-$, and $\mathcal{C}_0^*$ such that any (n, k, t)-bad vector $\mathbf{x}$ with $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{A}^+$ and $\mathcal{C}_{\mathbf{x}}^- \supseteq \mathcal{A}^-$ also has $\mathcal{C}_{\mathbf{x}}^+ \supseteq \mathcal{A}_0^+$, $\mathcal{A}_0^- \equiv \mathcal{A}^-$, and $\mathcal{C}_0^* = \binom{[n]}{k} \setminus (\mathcal{L}_k(\mathcal{A}_0^+) \cup \mathcal{R}_k(\mathcal{A}_0^-))$.

```
A_0^+, A_0^-, C_0^* ← PROPAGATENEGATIVE(n, k, t, A^+, A^-, C^*)
updated ← True
while updated do
  update ← False
  for all sets S ∈ C_0^* (in reverse lex order) do
    if P(n, k, A^+, A^- ∪ {S}) is infeasible then
      update ← True
      A_0^+ ← A_0^+ ∪ {S}.
      C_0^* ← C_0^* \ L_k(S)
      A_0^+, A_0^-, C_0^* ← PROPAGATENEGATIVE(n, k, t, A_0^+, A_0^-, C_0^*)
    end if
  end for
end while
return A_0^+, A_0^-, C_0^*
```

Given a set S in $\mathcal{C}^*$, we can test the linear program for feasibility when S is added to $\mathcal{A}^-$ or $\mathcal{A}^+$.

Observation 13. *If S is a k-set where $P(n, k, \mathcal{A}^+, \mathcal{A}^- \cup \{S\})$ is infeasible, then all vectors that are feasible solutions to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ are also feasible solutions to $P(n, k, \mathcal{A}^+ \cup \{S\}, \mathcal{A}^-)$.*

Observation 14. *If S is a k-set where $P(n, k, \mathcal{A}^+ \cup \{S\}, \mathcal{A}^-)$ is infeasible, then all vectors that are feasible solutions to $P(n, k, \mathcal{A}^+, \mathcal{A}^-)$ are also feasible solutions to $P(n, k, \mathcal{A}^+, \mathcal{A}^- \cup \{S\})$.*

Given these two lemmas, we could test every set in $\mathcal{C}^+$ and determine which sets should be added to $\mathcal{A}^+$ or $\mathcal{A}^-$. The PROPAGATEPOSITIVE algorithm (Algorithm 3) places as many right-most sets into $\mathcal{A}^+$ as possible using Lemma 13.

Since PROPAGATEPOSITIVE optimizes a linear program for every set in $\mathcal{C}^*$, this algorithm is too slow to be effective within BRANCHANDCUT. However, since PROPAGATEPOSITIVE includes a call to PROPAGATENEGATIVE after every addition to $\mathcal{B}^+$, it is possible that a single call to PROPAGATEPOSITIVE results with the linear program $\mathcal{P}(n, k, \mathcal{A}_0^+, \mathcal{A}_0^-)$ infeasible. Since PROPAGATENEGATIVE adds negative sets to $\mathcal{A}^-$, there are likely more sets $S \in \mathcal{C}^*$ where adding S to $\mathcal{A}^-$ leads to an infeasible linear program. Such sets are added to $\mathcal{A}^+$ by PROPAGATEPOSITIVE, leading to more sets which satisfy the conditions in PROPAGATENEGATIVE.

Using PROPAGATEPOSITIVE in place of PROPAGATENEGATIVE in the propagation step of BRANCHANDCUT is slow for even $k = 5$. However, for $k = 3$ and $k = 4$, using PROPA-

Algorithm 4 StochasticPropagation$(n, k, t, \mathcal{A}^+, \mathcal{A}^-, \mathcal{C}^*)$ — Randomly test sets for inclusion in $\mathcal{A}^-$ and $\mathcal{A}^+$.

Require: $k \geq 3$, $n \geq k$, $t \leq \binom{n-1}{k-1}$, and $\mathcal{B}^+, \mathcal{B}^- \subset \binom{[n]}{k}$.
Ensure: Sets added to $\mathcal{A}^+$ or $\mathcal{A}^-$ satisfy the hypotheses of Observations 13 or 14.

```
loop
  A+, A−, C* ← PropagateNegative(n, k, t, A+, A−, C*)   (see Algorithm 2)
  updated ← False
  while not updated do
    Randomly select a set S ∈ C*
    if P(n, k, A+, A− ∪ {S}) is infeasible then
      A+ ← A+ ∪ {S},   C* ← C* \ L_k(S),   updated ← True
      if |L_k(A+)| ≥ t then
        return A+, A−, C*
      end if
    else if P(n, k, A+ ∪ {S}, A−) is infeasible then
      A− ← A− ∪ {S},   C* ← C* \ R_k(S),   updated ← True
    end if
  end while
end loop
return A+, A−, C*
```

gatePositive terminates in at most three iterations of the loop and requires no branching. The log of this computation is small enough that we present the computation as proofs that $f(3) \leq 11$ and $f(4) \leq 14$ in Appendices B and C.

While PropagatePositive is unreasonably slow, PropagateNegative is very fast to test since we have very quick methods for computing $L_k^*(S)$. If we can find just one set S where adding S to $\mathcal{A}^-$ creates an infeasible linear program, then we can add S to $\mathcal{B}^+$ and likely the next iteration of PropagateNegative will add more negative sets.

Instead of carefully selecting a set S to peform this test, we simply select a set S in $\mathcal{C}^*$ uniformly at random. By randomly selecting sets and testing the linear program, we may quickly find a set that we can guarantee to be in $\mathcal{C}_{\mathbf{x}}^+$ for any (n, k, t)-bad vector $\mathbf{x} \in F_n$. This is the idea for the algorithm StochasticPropagation (Algorithm 4). Simply, we select a set from $\mathcal{C}^*$ at random and test if the set fits the hypotheses of Observations 13 or 14. We continue sampling until either (a) we find such a set and add it to $\mathcal{A}^+$ or $\mathcal{A}^-$, (b) we sample a specified number of sets which all fail these conditions, or (c) we reach a specified time limit. The sampling limits of (b) and (c) are not listed in Algorithm 4, but are both parameters which can be modified in our implementation.

StochasticPropagation is a zero-error randomized algorithm: it adds sets to $\mathcal{A}^+$ or $\mathcal{A}^-$ only when previous evidence guarantees that is the correct choice. The only effect of the randomness is how many sets actually are determined to be placed in $\mathcal{A}^+$ or $\mathcal{A}^-$.

What is particularly important is that the propagation in StochasticPropagation is stronger than PropagateNegative, but it is still slower. For the best performance,

we need a balance between the branching procedure and the propagation procedure. This balance is found by adjusting the number of random samples between successful updates in StochasticPropagation, as well as the total time to spend in that algorithm. Further, we can disable the call to StochasticPropagation until a certain number of k-sets have been added to $\mathcal{B}^+$ or $\mathcal{B}^-$, thereby strongly constraining the linear program and leading to a more effective random sampling process.

Solving the case $k = 7$ and $n = 22$ required more than 100 days of computation time for the deterministic branch-and-cut method, but required only a few hours using the randomized algorithm.

4.1 Results Using StochasticPropagation

Using BranchAndCut with StochasticPropagation, we computed $g(n, 7)$ for $8 \leq n \leq 29$ and verified that $f(k) = 3k + 2$ for $k = 7$.

Theorem 15. $f(7) = 23$.

The computation for this theorem is the same as in Thoerem 12, except we replace PropagateNegative with StochasticPropagation.

4.2 Implementation

The BranchAndCut algorithm was implemented in the *MMSConjecture* project within the *SearchLib* collection[2]. The software is available for download, reuse, and modification under the GNU Public License 3.0.

For $k \in \{4, 5\}$, we were able to verify the statements using the exact arithmetic solver supplied with GLPK [9], but for $k \geq 6$ this method was too slow and instead was verified using the floating-point linear programming software CPLEX [8]. The number of search nodes for each search, as well as the amount of computation time for each case are presented in Tables 1, 2, and 3. Empty cells refer to computations that were too long to complete (but only for cases where k divides n) and cells containing "—" refer to experiments reporting less than 0.01 seconds of computation time. In addition to the data presented here, all collected statistics and sharp examples are available online[3].

Execution times under one day were executed in a single process on a 2.3 GHz Intel Core i5 processor. Longer execution times are from parallel execution on the Open Science Grid [13] using the University of Nebraska Campus Grid [16]. The nodes available on the University of Nebraska Campus Grid consist of Xeon and Opteron processors with a range of speed between 2.0 and 2.8 GHz.

[2] *SearchLib* is available at http://www.math.illinois.edu/~stolee/SearchLib/.

[3] All data is available at http://www.math.illinois.edu/~stolee/data.htm.

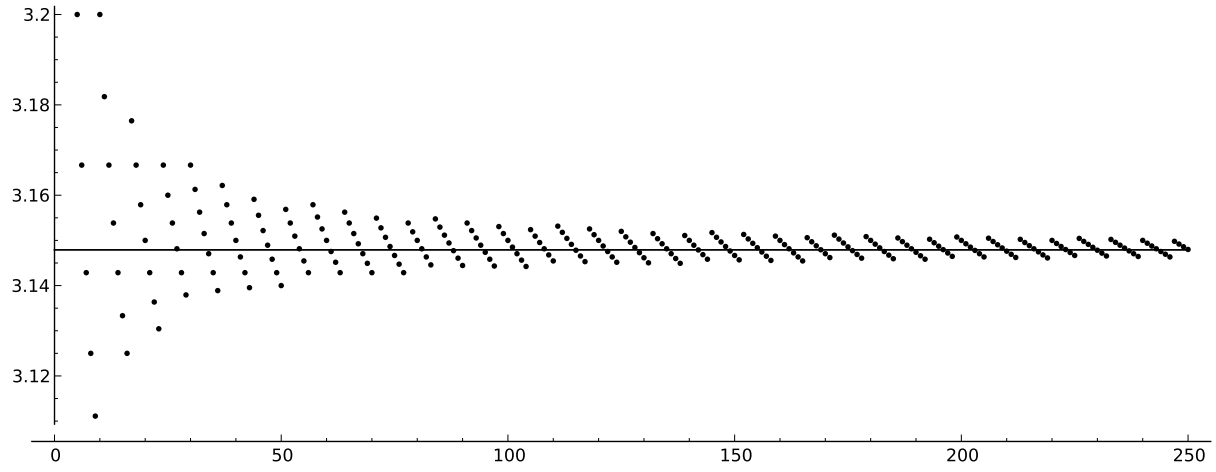

Figure 1: Values of N_k/k for $5 \leq k \leq 250$ and the line $y = \lim_{k\to\infty} N_k/k$.

5 Sharp Examples, Uniqueness, and the Strengthened Conjecture

In Tables 1, 2, and 3, the right-most column contains descriptions of the vectors $\mathbf{x}$ with $s_k(\mathbf{x})$ of minimum value. For $n \geq f(k)$, the vectors listed had $s_k(\mathbf{x})$ of minimum value while also having $\sigma_T(\mathbf{x}) < 0$ for $T = \{1, n-k+2, \ldots, n\}$. Observe that for all $n < f(k)$, the extremal examples use only two numbers, and have the form $a^b\ (-b)^a$ where $a + b = n$.

Recall Conjecture 2, where we claim $f(k)$ is exactly equal to N_k, where N_k is the minimum integer such that $\binom{N_k-3}{k} \geq \binom{N_k-1}{k-1}$. We make this conjecture for two reasons. First, the example of $3^{n-3}\ (-(n-3))^3$ was known to have fewer than $\binom{n-1}{k-1}$ nonnegative k-sums when $3k < n < N_k$ by previous authors (for example, see [7] where these vectors were used to show a lower bound $f(k) \geq \frac{22k}{7}$), but no examples were previously discovered for $n \geq N_k$. Second, our search for extremal vectors found no violation to this bound, but also showed that the extremal examples have the form $a^b\ (-b)^a$ when $n \leq N_k$ and k does not divide n. We therefore tested all vectors of the form $a^b\ (-b)^a$ for all $k \leq 250$. For all numbers n with $3k < n < N_k$ the example $3^{n-3}\ (-(n-3))^3$ was the best vector of this form, and for $N_k \leq n < 4k$ the best vector was $(n-1)^1\ (-1)^{n-1}$.

Figure 1 contains a plot of N_k/k and the line $y = \lim_{k\to\infty} N_k/k$, to demonstrate the conjectured values of $f(k)/k$.

Another strengthening of the conjecture that follows from our method (for $k \leq 7$) and that of Chowdhury (for $k = 3$) is that when $n \geq f(k)$, a vector $\mathbf{x} \in F_n$ has $s_k(\mathbf{x}) = \binom{n-1}{k-1}$ only if $x_1 + x_{n-k+2} + \cdots + x_n$ is nonnegative. Essentially, any sharpness example contains the same set of nonnegative k-sums as the sharpness example $(n-1)^1\ (-1)^{n-1}$.

We test this example is essentially unique by searching for an vector $\mathbf{x} \in F_n$ where $\sigma_T(\mathbf{x}) < 0$ for $T = \{1, n-k+2, \ldots, n\}$ and $s_k(\mathbf{x}) \leq \binom{n-1}{k-1}$. If no such vector is found, then the sharpness example is "unique," since all vectors $\mathbf{x} \in F_n$ with $s_k(\mathbf{x}) \leq \binom{n-1}{k-1}$ have

$\mathcal{C}_{\mathbf{x}}^{+} = \mathcal{L}_k(T)$. Define $g_s(n,k)$ to be the minimum $s_k(\mathbf{x})$ over all $\mathbf{x} \in F_n$ where $\sigma_T(\mathbf{x}) < 0$. We can use our target value t to find an (n,k,t)-bad vector $\mathbf{x} \in F_n$ with $\sigma_T(\mathbf{x}) < 0$. While the bound $t \leq \binom{n-1}{k-1}$ in the hypothesis of Lemma 6 does not hold, observe that since $\sigma_T(\mathbf{x}) < 0$ the conclusion of the lemma does hold in this scenario. Therefore, PROPAGATENEGATIVE (Algorithm 2) remains correct when verifying $g_s(n,k) \geq t$ using a call to BRANCHANDCUT$(n,k,t,\varnothing,\{T\},\binom{[n]}{k} \setminus \mathcal{R}_k(T))$.

Values of $g_s(n,k)$ were computed for $3 \leq k \leq 6$, $f(k) \leq n < f(k)+k$ and are given in Table 4. The sharp examples for vectors $\mathbf{x} \in F_n$ with $\sigma_T(\mathbf{x}) < 0$ are given in the final columns of Tables 1 and 2. Observe that the sharp examples make a "phase transition" at $n = 4k$, where the sharp examples have the form $a^b\ (-b)^a$ for all $n < 4k$, but then the examples with $n \geq 4k$ have at least three distinct values. This may be hinting to a deeper truth concerning the originally conjectured bound of $f(k) \leq 4k$.

5.1 Conclusions

We developed two methods for verifying $g(n,k) \geq t$, which we used to prove our strengthening of the Manickam-Miklós-Singhi conjecture for all $k \leq 7$, extending the previously best result of $k \leq 3$ [7]. Our branch-and-cut method BRANCHANDCUT uses a branching procedure which we prove will, in finite time, verify $g(n,k) \geq t$ or find a vector $\mathbf{x} \in F_n$ with $s_k(\mathbf{x}) < t$. Using the randomized propagation algorithm STOCHASTICPROPAGATION, we computed $f(k)$ and the values of $g(n,k)$ for all $k \leq 7$ and $k < n < f(k)+k$. Our implementations were not successful in extending our results to larger values of k due to a combination of large computation time and memory requirements.

Acknowledgments

The authors thank Ameera N. Chowdhury for many interesting discussions on this problem. The authors also thank Igor Pak for suggesting a computational approach which led to our branch-and-cut method.

References

[1] N. Alon, H. Huang, and B. Sudakov. Nonnegative k-sums, fractional covers, and probability of small deviations. *J. Combin. Theory, Ser. B* 103(3), 784–796 (2012)

[2] Zs. Baranyai, On the factorization of the complete uniform hypergraph. In A. Hajnal, R. Rado, V. T. Sós, eds. *Infinite and Finite Sets, Proc. Coll. Keszthely*, Colloquia Math. Soc. János Bolyai, 10, North-Holland, 91–107 (1975).

[3] A. Bhattacharya, On a conjecture of Manickam and Singhi. *Discrete Math.* 272, 259–261 (2003).

[4] T. Bier, N. Manickam, The first distribution invariant of the Johnson-scheme. *Southeast Asian Bull. Math.* 11(1), 61–68 (1987).

[5] G. Chiaselotti, On a problem concerning the weight functions. *Europ. J. Combinatorics* 23, 15–22 (2002).

[6] G. Chiaselotti, G. Infante, G. Marino, New results related to a conjecture of Manickam and Singhi. *Europ. J. of Combinatorics* 29, 361–368 (2008).

[7] A. N. Chowdhury, *Shadows and Intersections*. Ph.D. dissertation, University of California San Diego (2012).

[8] IBM ILOG CPLEX 11.0 User's Manual. *ILOG CPLEX Division*, Incline Village, NV, (2007).

[9] A. Makhorin, GLPK–GNU Linear Programming Kit. http://www.gnu.org/software/glpk/, (2008).

[10] N. Manickam and D. Miklós. On the number of nonnegative partial sums of a nonnegative sum. In *Combinatorics* (Eger, 1987), volume 52 of *Colloq. Math. Soc. János Bolyai*, pages 385–392. North-Holland, Amsterdam, (1988).

[11] N. Manickam and N. M. Singhi. First distribution invariants and EKR theorems. *J. Combin. Theory Ser. A* 48(1), 91–103, (1988).

[12] G. Marino, G. Chiaselotti, A Method to Count the Positive 3-Subsets in a Set of Real Numbers with nonnegative Sum. *Europ. J. Combinatorics* 23, 619–629 (2002).

[13] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, et al. The Open Science Grid. In *Journal of Physics: Conference Series*, volume 78, pages 12–57. IOP Publishing, (2007).

[14] R. P. Stanley, *Enumerative Combinatorics, Vol. 2*. Cambridge Studies in Advanced Mathematics 62 (2001).

[15] M. Tyomkyn, An improved bound for the Manickam-Miklós-Singhi Conjecture. *Europ. J. Combinatorics* 33, 27–32 (2012).

[16] D. J. Weitzel. *Campus Grids: A framework to facilitate resource sharing*. Masters thesis, University of Nebraska–Lincoln, (2011).

A Computation Data

k	n	$g(n,k)$	$\hat{g}(n,k)$	Nodes	GLPK	CPLEX	Strong Example
3	4	1	2	2	—	—	$1^3\ (-3)^1$
3	5	3	3	2	—	—	$3^2\ (-2)^3$
3	6	10	0	8	—	—	
3	7	10	5	2	—	—	$2^5\ (-5)^2$
3	8	16	5	2	—	—	$5^3\ (-3)^5$
3	9	28	0	2	—	—	
3	10	35	1	4	—	—	$3^7\ (-7)^3$
3	11	45	0	6	—	—	$7^4\ (-4)^7$
3	12	55	0	2	—	—	$7^5\ (-5)^7$
3	13	66	0	2	—	—	$4^9\ (-9)^4$
4	5	1	3	2	—	—	$1^4\ (-4)^1$
4	6	5	5	2	—	—	$1^5\ (-5)^1$
4	7	10	10	2	—	—	$5^2\ (-2)^5$
4	8	35	0	268	0.47 s	0.31 s	
4	9	35	21	4	—	—	$2^7\ (-7)^2$
4	10	70	14	28	0.11 s	0.03 s	$2^8\ (-8)^2$
4	11	92	28	10	0.07 s	0.01 s	$8^3\ (-3)^8$
4	12	165	0	50	0.35 s	0.11 s	
4	13	210	10	52	0.82 s	0.15 s	$3^{10}\ (-10)^3$
4	14	286	0	30	0.65 s	0.10 s	$10^4\ (-4)^{10}$
4	15	364	0	24	0.51 s	0.10 s	$11^4\ (-4)^{11}$
4	16	455	0	6	0.15 s	0.04 s	$35^1\ 3^{10}\ (-16)^5$
4	17	560	0	8	0.31 s	0.07 s	$38^1\ 4^{10}\ (-13)^6$
5	6	1	4	2	—	—	$1^5\ (-5)^1$
5	7	6	9	2	—	—	$1^6\ (-6)^1$
5	8	16	19	4	—	—	$3^5\ (-5)^3$
5	9	35	35	4	0.01 s	—	$7^2\ (-2)^7$
5	10	126	0				
5	11	126	84	10	0.10 s	0.03 s	$2^9\ (-9)^2$
5	12	246	84	92	2.61 s	0.26 s	$7^5\ (-5)^7$
5	13	405	90	234	14.22 s	1.08 s	$10^3\ (-3)^{10}$
5	14	550	165	44	3.37 s	0.44 s	$11^3\ (-3)^{11}$
5	15	1001	0	996		12.03 s	
5	16	1287	78	342	2.43 m	7.78 s	$3^{13}\ (-3)^{13}$
5	17	1820	0	702	5.48 m	23.26 s	$10^7\ (-7)^{10}$
5	18	2380	0	364	4.46 m	20.77 s	$14^4\ (-4)^{14}$
5	19	3060	0	192	3.64 m	16.54 s	$15^4\ (-4)^{15}$
5	20	3876	0	64	2.13 m	9.29 s	$79^1\ 19^1\ (-1)^{14}\ (-21)^4$
5	21	4845	0	64	3.09 m	13.53 s	$67^1\ 4^{13}\ (-17)^7$

Table 1: Data for branch-and-cut method with GLPK and CPLEX, for $k \in \{3, 4, 5\}$.

k	n	$g(n,k)$	$\hat{g}(n,k)$	Nodes	GLPK	CPLEX	Stochastic	Strong Example
6	7	1	5	2	—	—	—	$1^6\ (-6)^1$
6	8	7	14	2	—	—	—	$1^7\ (-7)^1$
6	9	28	28	6	0.01 s	—	0.03 s	$1^8\ (-8)^1$
6	10	70	56	32	0.20 s	0.05 s	0.25 s	$8^2\ (-2)^8$
6	11	126	126	10	0.09 s	0.03 s	0.05 s	$9^2\ (-2)^9$
6	12	462	0					
6	13	462	330	24	1.88 s	0.16 s	0.21 s	$2^{11}\ (-11)^2$
6	14	924	363	294	1.32 m	6.17 s	8.41 s	$2^{12}\ (-12)^2$
6	15	1705	297	12408	2.94 h	6.47 m	4.76 m	$12^3\ (-3)^{12}$
6	16	2431	572	1296	34.19 m	1.68 m	54.20 s	$13^3\ (-3)^{13}$
6	17	3367	1001	266	10.84 m	44.14 s	17.04 s	$14^3\ (-3)^{14}$
6	18	6188	0	183960		9.66 h		
6	19	8008	560	7262		47.46 m	13.74 m	$3^{16}\ (-16)^3$
6	20	11628	0	27696		4.58 h	2.32 h	$3^{17}\ (-17)^3$
6	21	15504	0	8932		2.48 h	1.11 h	$17^4\ (-4)^{17}$
6	22	20349	0	4622		2.06 h	59.81 m	$18^4\ (-4)^{18}$
6	23	26334	0	2378		1.66 h	38.51 m	$19^4\ (-4)^{19}$
6	24	33649	0	764		49.35 m		$33^1\ 1^{16}\ (-7)^7$
6	25	42504	0	744		1.15 h	26.62 m	$104^1\ 4^{16}\ (-21)^8$

Table 2: Data for the branch-and-cut method using GLPK and CPLEX, for $k = 6$.

k	n	$g(n,k)$	$\hat{g}(n,k)$	Nodes	Deterministic	Stochastic	Sharp Example
7	8	1	6	2	—	—	$1^7\ (-7)^1$
7	9	8	20	2	—	—	$1^8\ (-8)^1$
7	10	35	48	8	—	—	$7^3\ (-3)^7$
7	11	92	118	12	0.02 s	0.02 s	$3^8\ (-8)^3$
7	12	246	216	100	0.39 s	0.92 s	$5^7\ (-7)^5$
7	13	462	462		0.36 s	0.13 s	$11^2\ (-2)^{11}$
7	14	1716	0	26			
7	15	1716	1287	58	6.02 s	9.59 s	$2^{13}\ (-13)^2$
7	16	3432	1573	1562	4.97 m	15.40 m	$2^{14}\ (-14)^2$
7	17	6116	1892	26852	2.61 h	1.07 h	$12^5\ (-5)^{12}$
7	18	10296	2080	450772	3.44 d	1.02 h	$5^{13}\ (-13)^5$
7	19	14924	3640	28778	1.25 d	1.15 h	$16^3\ (-3)^{16}$
7	20	20944	6188	3615	8.51 h	28.80 m	$17^3\ (-3)^{17}$
7	21	38760	0				
7	22	50388	3876	795236	160.57 d	3.08 h	$3^{19}\ (-19)^3$
7	23	74613	0	13013*		8.09 d*	
7	24	100947	0	7870*		4.61 d*	
7	25	134596	0	6531*		3.85 d*	
7	26	177100	0	12718*		9.25 d*	
7	27	230230	0	30807*		19.18 d*	
7	28	296010	0	5564*		2.51 d*	
7	29	376740	0	6002*		3.38 d*	

* These node counts and CPU times are averages of at least three runs using StochasticPropagation.

Table 3: Completed computations for $k = 7$ using CPLEX.

k	3	3	3	4	4	4	4	5	5	5	5	5
n	11	12	13	14	15	16	17	17	18	19	20	21
$g(n,k)$	45	55	66	286	364	455	560	1820	2380	3060	3876	4845
$g_s(n,k)$	46	80	84	311	375	455	750	1946	2562	3165	4876	6097

k	6	6	6	6	6	6
n	20	21	22	23	24	25
$g(n,k)$	11628	15504	20349	26334	33649	42054
$g_s(n,k)$	12376	17136	21777	27303	39836	50456
Time	5.71 d	15.91 d	2.26 d	19.70 h	67.60 d	30.00 d

Table 4: Comparisons of $g(n,k)$ and $g_s(n,k)$ when $f(k) \leq n < f(k) + k$.

B A Computer-Generated Proof that $f(3) \leq 11$

The following proofs were created by executing BranchAndCut with propagation step PropagatePositive (Algorithm 3) and writing down the k-sums which are determined to be strictly negative or nonnegative.

Theorem 16. $g(3,11) = \binom{10}{2} = 45$.

Proof. The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 45 of nonnegative sets:

$$\begin{array}{lll} x_1{+}x_6 + x_{11} & x_1{+}x_8 + x_{10} & x_2{+}x_5 + x_{11} \\ x_2{+}x_6 + x_{10} & x_2{+}x_7 + x_9 & x_3{+}x_4 + x_{11} \\ x_3{+}x_5 + x_9 & x_3{+}x_7 + x_8 & x_4{+}x_6 + x_8 \end{array}$$

The following sums generated by PropagatePositive (Algorithm 3) must be nonnegative or else the associated linear program (Definition 7) becomes infeasible:

$$\begin{array}{lll} x_4{+}x_6 + x_7 & x_4{+}x_5 + x_8 & x_3{+}x_4 + x_{10} \end{array}$$

These positive sets now force at least 56 nonnegative 3-sums, and our target was 45 nonnegative 3-sums. ∎

Theorem 17. $g(3,13) = \binom{12}{2} = 66$.

Proof. The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 66 nonnegative sets:

$$\begin{array}{lll} x_1{+}x_5 + x_{13} & x_1{+}x_6 + x_{12} & x_1{+}x_7 + x_{11} \\ x_3{+}x_5 + x_{12} & x_3{+}x_6 + x_{10} & x_4{+}x_5 + x_{11} \\ x_4{+}x_7 + x_9 & & \end{array}$$

The associated linear program is infeasible. ∎

C A Computer-Generated Proof that $f(4) \leq 14$

The following proofs were created by executing BranchAndCut with propagation step PropagatePositive (Algorithm 3) and writing down the k-sums which are determined to be strictly negative or nonnegative.

Theorem 18. $g(4, 14) = \binom{13}{3} = 286$.

Proof. The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 286 nonnegative sets:

$x_1 + x_7 + x_{13} + x_{14}$	$x_1 + x_8 + x_{12} + x_{14}$	$x_1 + x_9 + x_{11} + x_{14}$
$x_2 + x_5 + x_{10} + x_{14}$	$x_2 + x_6 + x_9 + x_{14}$	$x_2 + x_6 + x_{10} + x_{13}$
$x_2 + x_8 + x_9 + x_{13}$	$x_3 + x_4 + x_{11} + x_{14}$	$x_3 + x_5 + x_9 + x_{14}$
$x_3 + x_5 + x_{10} + x_{13}$	$x_3 + x_6 + x_8 + x_{14}$	$x_3 + x_6 + x_9 + x_{13}$
$x_3 + x_6 + x_{10} + x_{12}$	$x_3 + x_7 + x_8 + x_{13}$	$x_3 + x_7 + x_9 + x_{12}$
$x_4 + x_5 + x_8 + x_{14}$	$x_4 + x_5 + x_9 + x_{13}$	$x_4 + x_6 + x_8 + x_{13}$
$x_4 + x_6 + x_9 + x_{12}$	$x_4 + x_8 + x_{10} + x_{11}$	$x_5 + x_7 + x_8 + x_{12}$
$x_5 + x_7 + x_{10} + x_{11}$	$x_6 + x_8 + x_9 + x_{11}$	

The following sums generated by PropagatePositive (Algorithm 3) must be nonnegative or else the associated linear program becomes infeasible:

$x_3 + x_4 + x_7 + x_9$	$x_3 + x_4 + x_6 + x_{10}$	$x_2 + x_4 + x_8 + x_{10}$
$x_1 + x_7 + x_8 + x_9$	$x_1 + x_6 + x_8 + x_{11}$	$x_1 + x_5 + x_9 + x_{11}$
$x_1 + x_5 + x_7 + x_{12}$	$x_1 + x_4 + x_{10} + x_{11}$	$x_1 + x_4 + x_9 + x_{12}$
$x_1 + x_4 + x_8 + x_{13}$	$x_1 + x_4 + x_5 + x_{14}$	$x_1 + x_2 + x_{10} + x_{13}$
$x_1 + x_2 + x_8 + x_{14}$		

These positive sets now force at least 199 nonnegative 4-sums.

The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 286 of nonnegative sets:

$x_1 + x_7 + x_{12} + x_{14}$	$x_1 + x_8 + x_{11} + x_{14}$	$x_2 + x_4 + x_{11} + x_{14}$
$x_2 + x_5 + x_9 + x_{14}$	$x_2 + x_5 + x_{11} + x_{13}$	$x_2 + x_6 + x_8 + x_{14}$
$x_2 + x_6 + x_9 + x_{13}$	$x_2 + x_7 + x_{10} + x_{12}$	$x_3 + x_4 + x_{10} + x_{14}$
$x_3 + x_4 + x_{12} + x_{13}$	$x_3 + x_5 + x_8 + x_{14}$	$x_3 + x_5 + x_9 + x_{13}$
$x_3 + x_5 + x_{10} + x_{12}$	$x_3 + x_6 + x_7 + x_{14}$	$x_3 + x_6 + x_8 + x_{13}$
$x_3 + x_6 + x_9 + x_{12}$	$x_3 + x_7 + x_8 + x_{12}$	$x_3 + x_7 + x_{10} + x_{11}$
$x_3 + x_8 + x_9 + x_{11}$	$x_4 + x_5 + x_7 + x_{14}$	$x_4 + x_5 + x_8 + x_{13}$
$x_4 + x_5 + x_9 + x_{12}$	$x_4 + x_6 + x_7 + x_{13}$	$x_4 + x_6 + x_8 + x_{12}$
$x_4 + x_6 + x_9 + x_{11}$	$x_5 + x_7 + x_8 + x_{11}$	

The following sums generated by PropagatePositive (Algorithm 3) must be nonnegative or else the associated linear program becomes infeasible:

$$\begin{array}{lll} x_3+x_5+x_6+x_9 & x_3+x_4+x_8+x_{10} & x_2+x_6+x_7+x_{10} \\ x_2+x_5+x_8+x_{10} & x_2+x_4+x_9+x_{10} & x_1+x_8+x_9+x_{10} \\ x_1+x_7+x_{10}+x_{11} & x_1+x_7+x_8+x_{13} & x_1+x_6+x_9+x_{12} \\ x_1+x_5+x_{10}+x_{13} & x_1+x_5+x_8+x_{14} & x_1+x_4+x_{10}+x_{14} \end{array}$$

These positive sets now force at least 265 nonnegative 4-sums.

The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 286 nonnegative sets:

$$\begin{array}{lll} x_1+x_5+x_{12}+x_{14} & x_1+x_6+x_{11}+x_{14} & x_1+x_7+x_{12}+x_{13} \\ x_1+x_8+x_{10}+x_{14} & x_1+x_8+x_{11}+x_{13} & x_2+x_3+x_9+x_{14} \\ x_2+x_3+x_{10}+x_{13} & x_2+x_4+x_7+x_{13} & x_2+x_4+x_9+x_{12} \\ x_2+x_5+x_6+x_{14} & x_2+x_5+x_8+x_{12} & x_2+x_6+x_9+x_{11} \\ x_3+x_4+x_6+x_{13} & x_3+x_4+x_8+x_{12} & x_3+x_5+x_7+x_{12} \\ x_3+x_5+x_8+x_{11} & x_3+x_6+x_7+x_{11} & x_3+x_7+x_9+x_{10} \\ x_4+x_5+x_6+x_{12} & x_4+x_5+x_7+x_{11} & x_4+x_6+x_8+x_{10} \\ x_5+x_7+x_8+x_9 & & \end{array}$$

The associated linear program is infeasible. ∎

Theorem 19. $g(4,15) = \binom{14}{3} = 364$.

Proof. The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 364 nonnegative sets:

$$\begin{array}{lll} x_1+x_8+x_{14}+x_{15} & x_1+x_9+x_{13}+x_{15} & x_1+x_{11}+x_{12}+x_{15} \\ x_2+x_5+x_{11}+x_{15} & x_2+x_6+x_{10}+x_{15} & x_2+x_6+x_{11}+x_{14} \\ x_2+x_7+x_9+x_{15} & x_2+x_7+x_{10}+x_{14} & x_2+x_8+x_{12}+x_{13} \\ x_2+x_{10}+x_{11}+x_{13} & x_3+x_4+x_{13}+x_{15} & x_3+x_5+x_9+x_{15} \\ x_3+x_5+x_{10}+x_{14} & x_3+x_6+x_8+x_{15} & x_3+x_6+x_9+x_{14} \\ x_3+x_6+x_{10}+x_{13} & x_3+x_8+x_9+x_{13} & x_3+x_9+x_{11}+x_{12} \\ x_4+x_5+x_{12}+x_{13} & x_4+x_6+x_9+x_{13} & x_4+x_7+x_8+x_{14} \\ x_4+x_7+x_{10}+x_{12} & x_5+x_6+x_8+x_{14} & x_5+x_8+x_9+x_{12} \end{array}$$

The following sums generated by PropagatePositive (Algorithm 3) must be nonnegative or else the associated linear program becomes infeasible:

$$\begin{array}{lll} x_3+x_4+x_7+x_{10} & x_3+x_4+x_6+x_{11} & x_2+x_4+x_7+x_{11} \\ x_1+x_7+x_9+x_{12} & x_1+x_6+x_8+x_{13} & x_1+x_5+x_{11}+x_{12} \\ x_1+x_5+x_8+x_{14} & x_1+x_4+x_{11}+x_{13} & x_1+x_4+x_7+x_{15} \\ x_1+x_3+x_8+x_{15} & x_1+x_2+x_{11}+x_{15} & \end{array}$$

These positive sets now force at least 267 nonnegative 4-sums.

The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have 364 nonnegative sets:

$x_1 + x_3 + x_{13} + x_{15}$
$x_1 + x_5 + x_8 + x_{15}$
$x_1 + x_6 + x_8 + x_{14}$
$x_1 + x_8 + x_{10} + x_{12}$
$x_2 + x_4 + x_8 + x_{14}$
$x_2 + x_5 + x_8 + x_{13}$
$x_2 + x_6 + x_{10} + x_{11}$
$x_3 + x_4 + x_{10} + x_{12}$
$x_3 + x_5 + x_8 + x_{12}$
$x_4 + x_5 + x_9 + x_{11}$
$x_5 + x_7 + x_9 + x_{10}$
$x_1 + x_4 + x_9 + x_{15}$
$x_1 + x_5 + x_9 + x_{14}$
$x_1 + x_6 + x_9 + x_{13}$
$x_2 + x_3 + x_9 + x_{15}$
$x_2 + x_4 + x_9 + x_{13}$
$x_2 + x_5 + x_9 + x_{12}$
$x_2 + x_7 + x_9 + x_{11}$
$x_3 + x_5 + x_6 + x_{15}$
$x_3 + x_5 + x_{10} + x_{11}$
$x_4 + x_6 + x_7 + x_{12}$
$x_1 + x_4 + x_{11} + x_{14}$
$x_1 + x_5 + x_{10} + x_{13}$
$x_1 + x_7 + x_{11} + x_{12}$
$x_2 + x_3 + x_{10} + x_{14}$
$x_2 + x_5 + x_7 + x_{14}$
$x_2 + x_6 + x_8 + x_{12}$
$x_3 + x_4 + x_7 + x_{15}$
$x_3 + x_5 + x_7 + x_{13}$
$x_3 + x_6 + x_8 + x_{11}$
$x_4 + x_8 + x_9 + x_{10}$

The associated linear program is infeasible. ∎

Theorem 20. $g(4, 17) = \binom{16}{3} = 560$.

Proof. The following sums generated by PropagateNegative (Algorithm 2) must be strictly negative or we have at least 560 nonnegative sets:

$x_1 + x_6 + x_{16} + x_{17}$
$x_1 + x_8 + x_{15} + x_{16}$
$x_2 + x_6 + x_{12} + x_{17}$
$x_2 + x_7 + x_{12} + x_{16}$
$x_2 + x_8 + x_{13} + x_{15}$
$x_3 + x_5 + x_{13} + x_{16}$
$x_3 + x_7 + x_9 + x_{17}$
$x_3 + x_8 + x_{10} + x_{15}$
$x_4 + x_5 + x_{10} + x_{17}$
$x_4 + x_6 + x_{11} + x_{15}$
$x_4 + x_7 + x_{10} + x_{15}$
$x_4 + x_8 + x_{10} + x_{14}$
$x_5 + x_6 + x_{10} + x_{15}$
$x_5 + x_7 + x_9 + x_{15}$
$x_6 + x_7 + x_{12} + x_{13}$
$x_1 + x_7 + x_{13} + x_{17}$
$x_1 + x_9 + x_{13} + x_{16}$
$x_2 + x_6 + x_{14} + x_{16}$
$x_2 + x_8 + x_{10} + x_{17}$
$x_2 + x_9 + x_{12} + x_{15}$
$x_3 + x_6 + x_{10} + x_{16}$
$x_3 + x_7 + x_{11} + x_{15}$
$x_3 + x_8 + x_{12} + x_{14}$
$x_4 + x_5 + x_{11} + x_{16}$
$x_4 + x_6 + x_{13} + x_{14}$
$x_4 + x_7 + x_{11} + x_{14}$
$x_4 + x_9 + x_{12} + x_{13}$
$x_5 + x_6 + x_{12} + x_{14}$
$x_5 + x_7 + x_{10} + x_{14}$
$x_7 + x_9 + x_{10} + x_{13}$
$x_1 + x_8 + x_{12} + x_{17}$
$x_2 + x_5 + x_{15} + x_{17}$
$x_2 + x_7 + x_{11} + x_{17}$
$x_2 + x_8 + x_{11} + x_{16}$
$x_3 + x_5 + x_{11} + x_{17}$
$x_3 + x_6 + x_{12} + x_{15}$
$x_3 + x_7 + x_{13} + x_{14}$
$x_3 + x_9 + x_{11} + x_{14}$
$x_4 + x_6 + x_9 + x_{17}$
$x_4 + x_7 + x_9 + x_{16}$
$x_4 + x_8 + x_9 + x_{15}$
$x_5 + x_6 + x_9 + x_{16}$
$x_5 + x_7 + x_8 + x_{17}$
$x_5 + x_8 + x_{11} + x_{13}$

The following sums generated by PropagatePositive (Algorithm 3) must be nonnegative or

else the associated linear becomes infeasible:

$$
\begin{array}{lll}
x_4 + x_6 + x_9 + x_{11} & x_4 + x_6 + x_7 + x_{13} & x_4 + x_5 + x_{10} + x_{11} \\
x_4 + x_5 + x_9 + x_{14} & x_4 + x_5 + x_6 + x_{15} & x_3 + x_7 + x_9 + x_{11} \\
x_3 + x_6 + x_9 + x_{13} & x_3 + x_6 + x_7 + x_{14} & x_3 + x_5 + x_{11} + x_{12} \\
x_3 + x_5 + x_{10} + x_{14} & x_3 + x_5 + x_7 + x_{15} & x_3 + x_4 + x_{11} + x_{13} \\
x_3 + x_4 + x_9 + x_{15} & x_3 + x_4 + x_6 + x_{16} & x_2 + x_8 + x_{10} + x_{12}
\end{array}
$$

These positive sets now force at least 560 nonnegative 4-sums, and our target was 560 nonnegative 4-sums. ∎