# COLUMN GENERATION OF POWER CONES FOR SIGNOMIAL OPTIMIZATION

A PREPRINT

**Anish Senapati**
CMS Department
California Institute of Technology
Pasadena, CA 91126
`asenapat@caltech.edu`

**Riley Murray**
Department of Electrical Engineering and Computer Sciences
UC Berkeley
Berkeley, CA
`rjmurray@caltech.edu`

**Adam Wierman**
CMS Department
California Institute of Technology
Pasadena, CA 91126
`adamw@caltech.edu`

August 6, 2022

## ABSTRACT

Certifying function nonnegativity is a prevalent problem in applied mathematics with applications in optimization and engineering. A new method for the certification of polynomial and signomial nonnegativity has been proposed that uses the sums of arithmetic-geometric exponentials or "SAGE" decompositions for verification. SAGE-based optimization techniques are traditionally done with a type of convex optimization called relative entropy programming. Recent research shows those same optimization problems can be approximated by power cone programming, an appealing alternative to resolve the numerical issues from the nonstandard REP program. We propose and optimize an efficient column generation approach to identify and add power cones into the nonnegativity cone of interest. This generation approach numerically exhibits sublinear convergence to the SAGE bound. The convergence of this method has no dependence on the number of variables in the signomial. Additionally, assuming certain uniqueness conditions of the constraints generated, we proved that the constraints are defined by so-called "sublinear circuits". We tested this column generation approach on problems in previous signomial programming literature and resolve numerical issues initially present with the relative entropy programming approach.

***Keywords*** Signomial · Convex Optimization · Column Generation · Power Cone Programming

## 1 Introduction

Signomial optimization and signomial programming are prevalent problems in applied mathematics with applications present in many areas of engineering including in aeronautics [1, 2, 3, 4, 5] , circuit design [6], and communications networks [7]. A signomial is a function $f$ defined by $\boldsymbol{x} \mapsto \sum_{i=1}^{m} c_i \exp(\boldsymbol{a}_i \cdot \boldsymbol{x})$ for $c \in \mathbb{R}^m$ and vectors $\boldsymbol{a}_i \in \mathbb{R}^n$. Generally, engineering applications frame signomial optimization as a minimization problem over a domain $X \in \mathbb{R}^n$ traditionally written as $f^* = \min_{x \in X} f(x)$. A related question one might ask of a signomial is if the coefficients $c_i$ are such that $f$ is nonnegative for a domain $X \in \mathbb{R}^n$. Such *nonnegativity certificates* also have a direct connection with optimization problems since any certificate that $f - \gamma$ is nonnegative for all $x \in X$ is equivalent to saying that $\gamma$ is a lower bound to $f$ for all $x \in X$.

For general functions, it is computationally intractable to determine function negativity. Even for signomial functions, certifying negativity can be shown to be NP-hard by reduction to matrix copositivity [8]. With this in mind, we instead seek for conditions for signomial nonnegativity which are tractable to verify. One such example is *geometric programming* where the coefficient vector $c$ is limited to $\mathbb{R}^n_{++}$ which can be transformed into a convex optimization problem by an appropriate change of variables. Unfortunately, the coefficient limitations of geometric programming results in many signomials models not falling under its its framework. Recently, Chandrasekaran and Shah proposed sums of arithmetic-geometric exponentials or "SAGE" decompositions, which use *relative entropy programming* to certify signomial nonnegativity in $\mathbb{R}^n$[9]. Since then, the structural properties of SAGE certificates have been further analyzed and the SAGE methodology has been generalized for certain feasible sets $X$[10, 11].

Despite the immense success of SAGE decompositions, the nonstandard nature of its associated relative entropy program results in certain numerical issues in practice. Recent research has focused on the ability to convert problems that are far beyond development and approximate them using more standard conic programming paradigms. For instance, Ye and Xie looked at the recasting mixed integer exponential conic programs using second order conic and polyhedral approximations to improve the scalability of solving such optimization programs [12]. Additionally, such standard conic programming paradigms offer significant outlook for theoretical analysis. For instance, Lindstrom was able to compute error bounds to a conic feasibility problem in the case of exponential cones [13]. Solvers like MOSEK have seen rapid improvement in their ability to handle such standard conic programs (power and exponential cones) in both scale and accuracy [14, 15]. This raises the question of using another type of convex optimization programming to efficiently compute SAGE decompositions without a relative entropy program.

Our main contribution is to extend SAGE decompositions further using a connection between SAGE certificate cones and the more elementary power cone. This *power cone programming* approach does not search over the entire cone of SAGE certificates but is numerically easier for solvers like MOSEK and CP-SCS. For compact sets $X \subset \mathbb{R}^n$, we develop an algorithm which iteratively generates power cones to approximate our SAGE cone. We also prove structural properties of the generated search space under mild conditions. The algorithm is also extended to fit previous hierarchies for signomial programming. We test our algorithm on randomly generated signomials and signomial optimization problems from previous literature.

### 1.1 Notations and Conventions

We denote the positive reals with $\mathbb{R}_+$. Similarly, we define $\mathbb{R}^m_+$ to be the set of vectors where each element is positive. We use $X$ to denote a subset of $\mathbb{R}^n$. In this paper, we will look at $X$ which are compact and convex. For a vector $\boldsymbol{a}$, we denote the ith element as $\boldsymbol{a}_i$ and the vector formed by deleting the ith element as $\boldsymbol{a}_i$. For a matrix $\boldsymbol{A}$, we similarly denote the ith row as $\boldsymbol{A}_i$ and the submatrix created by deleting the ith row as $\boldsymbol{A}_i$. We will use $\boldsymbol{e}_i$ to denote the $i$th basis vector and $\mathbf{1}$ as the vector with all 1's. We use $[n]$ as shorthand for the set $\{1, 2, ..., n\}$.

A set is conA set $K$ is called a cone if $\boldsymbol{x} \in K \Rightarrow \lambda \boldsymbol{x} \in K$ for all $\lambda \geq 0$. For a cone $K$, the dual cone is defined by $K^\dagger = \{\boldsymbol{y} : \boldsymbol{y}^\intercal \boldsymbol{x} \geq 0 \text{ for all } \boldsymbol{x} \text{ in } K\}$. A convex set $K$ induces a support function $\sigma_X(\boldsymbol{\lambda}) = \sup\{\boldsymbol{\lambda}^\intercal \boldsymbol{x} : \boldsymbol{x} \text{ in } K\}$. Finally, we denote $D$ as the relative entropy function, a convex function defined by $D(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^m \boldsymbol{u}_i \log(\boldsymbol{u}_i/\boldsymbol{v}_i)$ for $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^m_+$.

For an matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a vector $c \in \mathbb{R}^m$, we write $f = \mathrm{Sig}(\boldsymbol{A}, c) = \sum_{i=1}^m c_i \exp(\boldsymbol{A}_i \cdot \boldsymbol{x})$. Note that in comparison to the definition in Section 1, this implies the rows of $\boldsymbol{A}$ define the exponential terms (or monomials) of our signomial $f$. To provide an clear representation across all signomials $f$, we will construct $f$ such that $\boldsymbol{A}$ has $\boldsymbol{A}_1 = \mathbf{0}$. This corresponds to a constant term in the signomial. Given a matrix $\boldsymbol{A}$ and a set $X$, we define the nonnegativity cone:

$$\mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A}, X) \doteq \left\{ \boldsymbol{c} : \sum_{i=1}^m \boldsymbol{c}_i \exp(\boldsymbol{A}_i \cdot \boldsymbol{x}) \geq 0 \,\forall\, \boldsymbol{x} \in X \right\} = \{ \boldsymbol{c} : \mathrm{Sig}(\boldsymbol{A}, \boldsymbol{c}) \geq 0 \,\forall\, \boldsymbol{x} \in X \}.$$

Section 2 reviews relevant definitions and background regarding conditional SAGE certificates for signomial nonnegativity and the process of column generation.
Section 3 introduces the generation algorithm of power cones used to approximate the SAGE nonnegativity cone. The algorithm incorporates the basic idea of column generation along with structural properties of the SAGE cone in order to retrieve appropriate parameters for each power cone. We extend the use of the algorithm to the SAGE relaxation hierarchy of level (p; q; l) introduced in (Murray, 2019)[11].
Section 4 displays numerical results for the algorithm produced in section 3.
Section 5 has some concluding remarks along with a list of possible future avenues of work.

## 2 Definitions and Literature Review

Within this paper, we aim to solve optimization problems of the form:

$$(f,g)_X = \inf\{f(\boldsymbol{x}) : \boldsymbol{x} \in X, g(\boldsymbol{x}) \geq 0\} \tag{1}$$

where $f$ and $g$ are signomial functions from $\mathbb{R}^n$. Our overarching goal is to produce lower bounds $(f,g)_X^*$ to $(f,g)_X$. Note that as mentioned in Section 1, this can be viewed from the lens of nonnegativity by creating lower bounds to $(f,g)_X = \sup\{\gamma : f - \gamma \in \mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A}, X)\}$ with the constraints $g(\boldsymbol{x}) \geq 0$ incorporated to redefine our domain X. We introduce some definitions and concepts to understand the algorithms in this paper.

In Section 2.1, we review the basics of SAGE certificates and discuss relevant structural properties of SAGE signomials in Section 2.2. In Section 2.3, we discuss the (p, q, l)-hierarchies for signomial programming which use modulators to improve the largest SAGE-certifiable bound on Eq. 1. Finally, in section 2.4 we introduce some of the terminology and concepts behind column generation, originally used to solve large mixed integer linear programming which will be employed in our SAGE certification.

### 2.1 Sum of AM-GM Exponentials

To create a tractable verification of nonnegativity for a coefficient vector $\boldsymbol{c} \in \mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A})$, we need some additional structure on the nature of the signomials in Equation (1). Chandresakaran and Shah focused on the case where the coefficient vector $\boldsymbol{c}$ contained at most one nonnegative entry $c_k$ [9]. A signomial $f = Sig(\boldsymbol{A}, \boldsymbol{c})$ is called X-AGE if it is nonnegative on $X$ and at most one monomial has a negative coefficient. The $k^{th}$ AGE cone of coefficients for $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is defined as the cone

$$\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, k, X) = \{\boldsymbol{c} : \boldsymbol{c}_{\setminus k} \geq 0 \text{ and } \boldsymbol{c} \text{ belongs to } \mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A}, X)\}$$

A signomial $f = Sig(\boldsymbol{A}, \boldsymbol{c})$ is called X-SAGE if it can be written as the sum of X-AGE signomials. On a similar vein, we define the cone of SAGE signomials coefficients for $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ as

$$\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X) = \left\{ \boldsymbol{c}, \boldsymbol{c} = \sum_{i=1}^{m} \boldsymbol{c_i} \text{ and } \boldsymbol{c_i} \in \mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X) \right\}$$

More briefly, we can write this as a Minkowski sum $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X) = \sum_{i=1}^{m} \mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$. This cone of X-SAGE coefficients now allows us to provide a bound for $(f,g)_X$ since $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X) \subseteq \mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A}, X)$ which implies that

$$(f,g)_X = \sup\{\gamma : \boldsymbol{c} - \gamma \boldsymbol{e}_1 \text{ in } \mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A}, X)\} \geq \sup\{\gamma : \boldsymbol{c} - \gamma \boldsymbol{e}_1 \text{ in } \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)\} =: (f,g)_X^{\mathsf{SAGE}}.$$

For computational purposes, we require a way to represent the cones $\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, k, X)$ and $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)$ respectively. The following two results give a efficient way to represent signomials in these cones.

**Theorem 1** (Murray, Chandrasekaran, & Wierman (2019) [11])**.** *Let $X \in \mathbb{R}^n$ be convex with support function $\sigma_X(\lambda)$. For a matrix $\boldsymbol{A} \in R^{m \times n}$, and index $k$ in $[m]$, we have*

$$\begin{aligned} \mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, k, X) = \{\boldsymbol{c} \in R^m : &\textit{ there exists } \boldsymbol{\nu} \in \mathbb{R}^{m-1} \textit{ and } \boldsymbol{\lambda} \textit{ in } R^n \\ &\textit{that satisfy } [\boldsymbol{A}_{\setminus k} - \boldsymbol{1}\boldsymbol{A}_k]^\dagger \boldsymbol{\nu} + \boldsymbol{\lambda} = 0 \\ &\textit{and } \sigma_X(\lambda) + D(\boldsymbol{\nu}, e\boldsymbol{c}_{\setminus k}) \leq \boldsymbol{c}_k\} \end{aligned}$$

The constraints in Thereom 1 are convex in $\boldsymbol{c}$ and $\nu$, so if X is a convex set, then $\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, X)$ is also a tractable convex set with inclusion tested by the *relative entropy program* above. Combined with the following theorem, this enables efficient procedures to search for SAGE certifications.

**Theorem 2** (Murray, Chandrasekaran, & Wierman (2019) [11])**.** *If $\boldsymbol{c} \in \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)$ with $N = \{\beta : c_\beta < 0\}$, there exists $\{\boldsymbol{c}^{(i)}\}_{i \in N}$ such that*

$$\boldsymbol{c}^{(i)} \in \mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, \beta, X),\ \boldsymbol{c} = \sum_{\beta \in N} \boldsymbol{c}^{(\beta)},\ \boldsymbol{c}^{(\beta)}_\beta = \boldsymbol{c}_\beta,\ \boldsymbol{c}^{(\beta)}_\alpha = 0 \textit{ for all } \alpha \neq \beta \in N$$

Practically, this structure preservation gives us a substantial presolve and optimization procedure for searching for SAGE certificates reducing the number of relative entropy programs from $O(m^2)$ to $O(km)$ where $k = |N|$. Combining the results of these two theorems allows us to construct representations of X-SAGE decompositions into $\{(\boldsymbol{\nu}^{(i)}, \boldsymbol{c}^{(i)})\}_{i=1}^k$ where $\boldsymbol{c}^{(i)} \in \mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$ and $\boldsymbol{\nu}^i$ are the corresponding witnesses of $\boldsymbol{c}^{(i)}$ from Theorem 1. This decomposition process is implemented in the python package SAGEOPT.

### 2.2 Structural Results of SAGE Certificates

In this section, we review some structural results regarding $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)$. We first introduce the set $\mathsf{N}_i = \{\boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{v}_{\setminus i} \geq 0, 1^\intercal \boldsymbol{v} = 0\}$. We denote the set of normalized vectors, $\boldsymbol{\lambda} = \boldsymbol{\nu}/\|\boldsymbol{\nu}\|_\infty$ for $\boldsymbol{\nu} \in \mathsf{N}_i$, as $\overline{\mathsf{N}}_i$. These normalized vectors have exactly one negative component $\boldsymbol{\lambda}_i = -1$ and their nonnegative components sum to 1.

We define a vector $\boldsymbol{v} \in R^m$ as an *X-circuit* of $\boldsymbol{A}$ if it is (1) nonzero, (2) $\sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{v}) < \infty$, and (3) it cannot be written as a convex combination of two linear independent vectors $\boldsymbol{v}^1, \boldsymbol{v}^2 \in \mathsf{N}_i$ for which $v \Rightarrow \sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{v})$ is linear on $[\boldsymbol{v}^1, \boldsymbol{v}^2]$. Using these definition, we can create the associated set $\Lambda_X(\boldsymbol{A}, i)$ for each $i \in [n]$:

$$\Lambda_X(\boldsymbol{A}, i) := \{\boldsymbol{\lambda} : \boldsymbol{\lambda} \in \overline{\mathsf{N}}_i, \lambda \text{ is an X-circuit of } \boldsymbol{A}\}$$

For ease of communication, we also define the sets $\boldsymbol{v}^+ := \{i, v_i > 0\}$ and $v^- := i$ such that $v_i < 0$. Further, to simpilify further notation, we define the functional form of an X-circuit as

**Definition 2.1.** *The functional form of an X-circuit $\boldsymbol{v} \in \mathbb{R}^m$ is $\phi_{\boldsymbol{v}} : \mathbb{R}^{m-} \Rightarrow \mathbb{R}$ defined by*

$$\phi_{\boldsymbol{v}}(\boldsymbol{y}) = \sum_{i \in [m]} \boldsymbol{y}_i \boldsymbol{v}_i + \sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{v}).$$

For $\lambda \in \overline{\mathsf{N}_i}$, we define the following two cones

**Definition 2.2.** *The primal power cone and its conic dual, the dual power cone are the sets*

$$Pow(\lambda) = \left\{ z : \prod_{\alpha \in \lambda^+} z_\alpha^{\lambda_\alpha} \geq |z_\beta|, z_{\setminus \beta} \geq 0 \right\}, \ \ Pow(\lambda)^\dagger = \left\{ w : \prod_{\alpha \in \lambda^+} \frac{w_\alpha}{\lambda_\alpha}^{\lambda_\alpha} \geq |w_\beta|, w_{\setminus \beta} \geq 0 \right\}$$

Finally, we also define the following cone and its conic dual:

**Definition 2.3.** *Given $\boldsymbol{\lambda} \in \overline{N_i}$, the $\boldsymbol{\lambda}$-witnessed AGE cone is*

$$\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X) := \left\{ \boldsymbol{c} : \prod_{\alpha \neq i} \left( \frac{\boldsymbol{c}_\alpha}{\boldsymbol{\lambda}_\alpha} \right)^{\boldsymbol{\lambda}_\alpha} \geq -\boldsymbol{c}_i \exp(\sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{\lambda})) \text{ and } \boldsymbol{c}_{\setminus i} \geq 0 \right\}$$

**Proposition 1.** *Given $\boldsymbol{\lambda} \in \overline{N_i}$, the dual $\boldsymbol{\lambda}$-witnessed AGE cone is given by*

$$\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X)^\dagger = \left\{ \boldsymbol{v} : \exp(\sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{\lambda})) \prod_{\alpha \neq i} \boldsymbol{v}_\alpha^{\lambda_\alpha} \geq \boldsymbol{v}_i \text{ and } \boldsymbol{v} > 0 \right\}$$

Proven by Murray [16], it is seen that with a diagonal linear operator $S_\lambda$ where $(S_\lambda \boldsymbol{c})_i = \boldsymbol{c}_i$ for $i \in \boldsymbol{c}^+$ and $(S_\lambda \boldsymbol{c})_i = \boldsymbol{c}_\beta \exp(\sigma_X(-\boldsymbol{A}^\dagger \lambda))$ for $i = \lambda_i$, there is an explicit connection between the primal $\lambda$-witnessed AGE cone and dual power cones.

**Proposition 2.** *For $\boldsymbol{\lambda} \in \overline{\mathsf{N}}_i$ and $\sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{\lambda}) < \infty$, the $\lambda$-witnessed age cone admits the representation*

$$\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X) = \{\boldsymbol{c} \in R^m | \boldsymbol{c}_{\setminus i} \geq 0 \text{ and } (S_\lambda \boldsymbol{c} - r\delta_i) \in Pow(\lambda)^\dagger \text{ for some } r \geq 0\} \tag{2}$$

There are two important results regarding $\mathsf{C}(\boldsymbol{A}, i, \boldsymbol{\lambda}, X)$. First, it can be shown that for $c \in \mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X)$ for any $\lambda$, $c$ is also in $\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$. Secondly, the above definitions of X-circuits and $\lambda$-witnessed AGE cones are related to SAGE signomials in the following theorem proven by Murray [16].

**Theorem 3.** *The cone $\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$ can be written as the convex hull of $\lambda$-witnessed AGE cones. In particular,*

$$\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X) = \operatorname{conv} \left( \bigcup_{\boldsymbol{\lambda} \in \Lambda_X(\mathsf{A}, \mathsf{i})} \mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X) \right). \tag{3}$$

This theorem states that the union of a specific set $\Lambda_X(\boldsymbol{A}, i) \in \overline{N_i}$ completely recovers $\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$. However, since $\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X) \subset \mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$ for all $\boldsymbol{\lambda}$, there is no harm in adding other $\lambda$-witnessed AGE cones into this union. Indeed, if desired, we could also say that

$$\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X) = \operatorname{conv} \left( \bigcup_{\lambda \in \overline{\mathsf{N}_i}} \mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X) \right) \tag{4}$$

### 2.3 (p, q, l) Hierarchy for Signomial Optimization

In this section, we review a SAGE based hierarchy of convex relaxations for signomial programming. For generality, we slightly edit our objective function to solve $(f, g, \phi)_X = \inf\{f(\boldsymbol{x}) : \boldsymbol{x} \in X, g(\boldsymbol{x}) \geq 0, \phi(\boldsymbol{x}) = 0\}$. This hierarchy serves to strengthen the lower bounds created by the SAGE certificates and build successively stronger bounds to $(f, g, \phi)_X$

Let $\boldsymbol{A}$ be the matrix of exponents such that $f$, $g$, and $\phi$ are all defined signomials over $\boldsymbol{A}$ and $\boldsymbol{A}$ is the smallest monomial basis to span $f$, $g$, and $\phi$. This SAGE relaxation is parametrized by 3 integers $(p, q, l)$ with $p \geq 0$ and $q \geq 1$. Denoting $\boldsymbol{A}[p]$ as the matrix of exponents for $Sig(\boldsymbol{A}, 1)^p$, $g[q]$ as the set of all products of at-most-q elements of g, and $\phi[q]$ similarly, we can construct the *(p, q, l)* level SAGE relaxation of $(f, g, \phi)_X$ to be:

$$\begin{aligned} (f,g,\phi)_X^{(p,q,l)} = \sup \gamma \textit{ such that } & s_h, z_h \text{ are signomials defined by } \boldsymbol{A}[p] \\ & L = f - \gamma - \sum_{h \in g[q]} s_h \cdot h - \sum_{h \in \phi[q]} z_h \cdot h \\ & \mathrm{Sig}(\boldsymbol{A}, \mathbf{1})^l L \in \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}[p], X) \\ & s_h, z_h \in \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}[p], X) \end{aligned} \tag{5}$$

### 2.4 Column Generation

In this section, we review the basics of column generation. Column generation is an algorithm originally developed to solve large linear programs where the number of variables is too large to consider them all explicitly. In these situations we can solve approximations of a linear program by only allowing a few per-determined variables $x_i > 0$ and forcing all other $x_j = 0$ where $x$ is the decision variable of interest. To properly set the stage, suppose that we have a linear program

$$\min_{\boldsymbol{x}} \{\boldsymbol{c}^\intercal \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\} \tag{6}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a wide matrix with $n$ being so large we cannot afford to solve the entire linear program. Let's say that we have an index set $I \subset [n]$ that is not too big. Then we can solve the following problem with no difficulties.

$$\mathrm{Val}(I) = \min\{\boldsymbol{c}_I^\intercal \boldsymbol{x}_I \;:\; (\boldsymbol{A}[:, I])\boldsymbol{x}_I = \boldsymbol{b}, \boldsymbol{x}_I \in \mathbb{R}_+^I\} \tag{7}$$

Once we solve (7), we extend $\boldsymbol{x}_I \in \mathbb{R}_+^I$ to a full vector $\boldsymbol{x} \in \mathbb{R}^n$ by setting $x_i = 0$ for all $i \in [n] \setminus I$.

Column generation can be attributed to 2 steps. First, we solve $\boldsymbol{x}$ for (7) and check to see if they solve the original problem in (6). In the case of the linear program in (6), it can be shown that if $(\boldsymbol{x}, \boldsymbol{y})$ solve (7) and its respective dual, then $\boldsymbol{x}$ solves (6) iff $\boldsymbol{y}$ is dual feasible. So, the termination condition for a linear program would be $\boldsymbol{c} - A^\intercal \boldsymbol{y} \geq 0$.

Second, if such a termination condition is not satisfied, we find a way to increase $I$ and retry solving (7). One approach would be to see what indices $j \in [n]$ satisfy $\boldsymbol{c}_j - \boldsymbol{A}[:, j]\boldsymbol{y} < 0$ and perform the update $I = I \cup \{j\}$ which is guaranteed to improve the bound $\mathrm{Val}(I)$. However, it is important to note that the algorithm to use to update out indices needs to have no dependence on $n$ since our premise was $n$ was too large to perform an algorithm on all $n$ columns directly. In both the termination condition and the update procedure just described, checking the sign for $\boldsymbol{c}_j - \boldsymbol{A}[:, j]\boldsymbol{y}$ would require iterating through all columns of A which is inefficient. Generally, we need to devise a more "efficient" algorithm to determine which indices to include. Such an algorithm is called a *separation oracle*. There is no universal separation oracle for optimization programs. Separation oracles have to be designed based on the structure of the problem at hand.

## 3 Column Generation Applied to X-SAGE Certificates

There are numerous optimization problems where the relative entropy program in Theorem 1 runs into numerical issues with solvers like MOSEK. With the use of Theorem 3, we can represent AGE cones with power cones, a more common conic paradigm. However, Theorem 3 relies on $\Lambda_X(\boldsymbol{A}, i)$ to completely recover the AGE cone of interest. The task of actually finding $\Lambda_X(\boldsymbol{A}, i)$ is a difficult problem even for a polyhedral domain $X$, let alone a more general convex $X$. Since every $\lambda$-witnessed AGE cone can be added into our union as in Equation (4), one idea might be to include all $\lambda \in \overline{\mathsf{N}}_i$ and optimize over this union. However, the number of normalized vectors in $\overline{\mathsf{N}}_i$ is too large to feasibly optimize over. So, it is reasonable to make a compromise to work with a smaller collection of $\lambda$-witnessed X-AGE cones $\Lambda_i \subset \overline{\mathsf{N}}_i$. Since $\Lambda_i$ might not contain all the vectors in $\Lambda_X(\boldsymbol{A}, i)$, such a subset might not fully recover

$\mathsf{C}_{\mathsf{AGE}}(\boldsymbol{A}, i, X)$. However, under appropriate choices of $\Lambda_i$, the Minkowski sum of these $\lambda$-witnessed AGE cones still serve as a good approximation of the original SAGE cone while being numerically easier from a solvers perspective.

In this section, we describe an iterative algorithm to generate $\lambda$ to include in our collections $\Lambda_i$. This algorithm is inspired from the ideas of column generation and the SONC generation algorithm introduced by Papp [17]. We focus our attention to a simpler signomial optimization problem in Section 3.1 and introduce our generation algorithm. We discuss initialization, convergence, and some optimization procedures developed when making the algorithm. In Section 3.2, we discuss an interesting property about this generation algorithm with relation to X-circuits. In section 3.3, we extend our $\lambda$ generation algorithm to the (p, q, l)-hierarchy of X-SAGE introduced in Section 2.3

### 3.1 Generating $\lambda$-witnessed AGE cones

In this section, we focus on the algorithm as applied to the problem of $f_X = \sup\{\gamma : c - \gamma\delta_1 \in \mathsf{C}_{\mathsf{NNS}}(\boldsymbol{A}, X)\}$. This can be viewed as the (0, 1, 0)-hierarchy of SAGE relaxations where the inequalities $g(\boldsymbol{x}) \geq 0$ are incorporated into our domain $X$. We denote the SAGE bound as $f_X^{\mathsf{SAGE}} = \sup\{\gamma : c - \gamma\delta_1 \in \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)\}$. Suppose we have finite sets $\Lambda_i \subset \overline{\mathsf{N}}_i$ and define $\Lambda = \cup_{i=1}^m \Lambda_i$. For any such set $\Lambda$ we define the cone:

$$\mathsf{C}_{\boldsymbol{\Lambda}}(\boldsymbol{A}, X) = \sum_{i=1}^{m} \sum_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}_i} \mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X) \tag{8}$$

This cone contains the coefficient vector of some $X$-SAGE signomials supported on $\boldsymbol{A}$. So, for a given signomial $f(\boldsymbol{x}) = \boldsymbol{c}^{\intercal} \exp \boldsymbol{A}\boldsymbol{x}$, we have

$$f_X^{\Lambda} := \sup\{\gamma \,:\, \boldsymbol{c} - \gamma\boldsymbol{\delta}_1 \in \mathsf{C}_{\boldsymbol{\Lambda}}(\boldsymbol{A}, X)\} \leq \sup\{\gamma \,:\, \boldsymbol{c} - \gamma\boldsymbol{\delta}_1 \in \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)\} = f_X^{\mathsf{SAGE}}. \tag{9}$$

The column generation algorithm in Algorithm 1 approximates $f_X^{\mathsf{SAGE}}$ with $f_X^{\Lambda}$ for progressively larger sets $\Lambda$.

**Algorithm 1** Column Generation of $\lambda$-witnessed AGE cones

**procedure** COLUMN_GENERATION_ITERATION($\Lambda, c \in \mathbb{R}^m, \boldsymbol{A} \in R^{m\times n}, X$)
  Solve and store the solution to the dual problem $f_X^{\boldsymbol{\Lambda}^\dagger}$ and obtain the dual variable $\boldsymbol{v}$ from the solution
  Remove $\lambda \in \Lambda$ which correspond to nonbinding constraints
  Set $y \leftarrow \log(v)$
  **for** $i \in [m]$ **do**
    Set $val_i \leftarrow \min_{\boldsymbol{\lambda}}\{\boldsymbol{y}^{\intercal}\boldsymbol{\lambda} + \sigma_X(-\boldsymbol{A}^{\intercal}\boldsymbol{\lambda})\,;\, \lambda \in \overline{\mathsf{N}_i}\}$
  **if** $val_i \geq 0$ for all $i$ **then**
    Return $\Lambda$, $f_X^{\boldsymbol{\Lambda}^\dagger}$, TRUE ▷ Column Generation Terminated
  **else**
    Update $\Lambda \leftarrow \Lambda \cup \{min_{j\in[n]} val_j\}$
    Return $\Lambda$, $f_X^{\boldsymbol{\Lambda}^\dagger}$, False ▷ Column Generation found another $\lambda$
**procedure** COLUMN_GENERATION_RUN($\Lambda_{init}, c \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{m\times n}, X$)
  $\Lambda \leftarrow \Lambda_{init}$, vals $\leftarrow \{\}$, status $\leftarrow$ FALSE
  **while** status $\leftarrow$ FALSE **do**
    $\Lambda$, current_val, status $\leftarrow$ Column_Generation_Iteration($\Lambda, c, \boldsymbol{A}, X$)
    vals $\leftarrow$ vals $\cup\{$current_val$\}$
  Return vals, $\Lambda$

We require solving the dual problem $f_X^{\boldsymbol{\Lambda}^\dagger}$ and recovering the dual variable $\boldsymbol{v}$. The dual of the problem $f_X^{\Lambda}$ can be stated as: $\{\inf \boldsymbol{c}^T\boldsymbol{v} : \boldsymbol{v}_1 = 1, \boldsymbol{v} \in \mathsf{C}_{\boldsymbol{\Lambda}}(\boldsymbol{A}, X)^\dagger\}$ where $\mathsf{C}_{\boldsymbol{\Lambda}}(\boldsymbol{A}, X)^\dagger$ is the dual cone of the Minkowski sum in Eq. (8). The dual cone can be implemented with $\mathsf{C}_{\boldsymbol{\Lambda}}(\boldsymbol{A}, X) = \cap_{\lambda\in\Lambda}\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X)^\dagger$ and dual lambda-witnessed AGE cones $\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X)^\dagger$ come from Propostion 1. Since we work in the dual cone space, it is perhaps more appropriate to refer to this algorithm as *constraint generation* since we are generating constraints for which $\boldsymbol{v}$ must be constrained to in the dual space.

A few statements can be stated about the algorithm in regards to its correctness and convergence.

**Proposition 3.** *If the algorithm terminates, then* $f_X^{\boldsymbol{\Lambda}\dagger} = f_X^{\mathsf{SAGE}}$

*Proof.* Suppose for the sake of contradiction that they are not equal. Since strong duality holds in the computation of $f_X^{\mathsf{SAGE}}$, we have that $f_X^{\mathsf{SAGE}} = f_X^{\mathsf{SAGE}\dagger} \neq f_X^{\mathbf{\Lambda}\dagger}$. Note that since $\mathsf{C}_{\mathbf{\Lambda}}(\boldsymbol{A}, X) \subseteq \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)$, order reversal of dual cones implies that $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)^\dagger \subseteq \mathsf{C}_{\mathbf{\Lambda}}(\boldsymbol{A}, X)^\dagger$. So, looking at the structure of the problems $f_X^{\mathbf{\Lambda}\dagger}$ and $f_X^{\mathsf{SAGE}\dagger}$, it follows with our assumption that $\boldsymbol{v} \in \mathsf{C}_{\mathbf{\Lambda}}(\boldsymbol{A}, X)^\dagger$ that optimizes $f_X^{\mathbf{\Lambda}\dagger}$ in Step 1 is not in $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)^\dagger$. Using Murray's [16, Theorem 5.5.4], we know that any $\boldsymbol{v} = \exp \boldsymbol{y}$ that belongs to $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)^\dagger$ if and only if $\phi_\lambda(y) \geq 0$ for all $\lambda$ in $\Lambda_X(\boldsymbol{A})$. Since our termination condition checks that $val_i \geq 0$ for all $i$, we have that $\phi_\lambda(y) \geq 0$ for all $\lambda \in \Lambda_X(\boldsymbol{A}) \subset \cup\{\overline{N_i}\ \forall i \in [m]\}$. So, the dual variable $\boldsymbol{v}$ must be contained in $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)^\dagger$, contradicting the previous statement of $\boldsymbol{v}$ and proving our claim. $\square$

We will explain one nonterminal iteration of the algorithm. Firstly, note that [16, Lemma 5.5.12] can be modified to: If $\boldsymbol{y} \in \mathbb{R}^n$ satisfies $\phi_\lambda(\boldsymbol{y}) < 0$ for some $\lambda \in \overline{\mathsf{N}}_i$, then $\exp \boldsymbol{y} \notin \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)$. This change is valid since the construction provided in the proof of the lemma does not use any significant property of X-circuits beyond them being contained in $\overline{\mathsf{N}}_i$ for some $i \in [m]$.

Consider one nonterminal iteration of the constraint generation algorithm. We are guaranteed that the $\boldsymbol{\lambda}$ added to $\Lambda$ satisfies $\phi_\lambda(\boldsymbol{y}) < 0$ for $\boldsymbol{y} = \log(\boldsymbol{v})$. Using the modification of [16, Lemma 5.5.12], this implies that $\boldsymbol{v} \notin \mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)^\dagger$. So, at each iteration, we find a $\boldsymbol{v} \in \mathsf{C}_{\mathbf{\Lambda}}(\boldsymbol{A}, X)^\dagger$ that is not contained $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)^\dagger$. Now, exponentiating $\phi_\lambda(\boldsymbol{y}) < 0$ along with the fact that $\boldsymbol{\lambda} \in \overline{\mathsf{N}}_i \implies \boldsymbol{\lambda}_i = -1$ leads to the fact that

$$\exp(\sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{\lambda})) \prod_{\alpha \neq i} \boldsymbol{v}_\alpha^{\lambda_\alpha} < \boldsymbol{v}_i,$$

exactly the opposite of the constraint defined by $\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X)^\dagger$. Subsequently adding the constraint of $\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, i, X)^\dagger$ will remove $\boldsymbol{v}$ from the feasible set given by $\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, X)^\dagger$. This entails that the optimal value for $f_X^{\mathbf{\Lambda}^\dagger}$ at iteration $k+1$ is at least as good as the optimal value at iteration $k$ for any $k$. Combining this with Proposition 3, we can establish that the limit of the sequence $\{f_X^{\mathbf{\Lambda}\dagger}\}$ is indeed $f_X^{\mathsf{SAGE}}$. It is important to note that while any nonterminal iteration eliminates $\boldsymbol{v}$ from the the feasible set, it is not guaranteed that each iteration has a strictly increasing value for $\{f_X^{\mathbf{\Lambda}\dagger}\}$. Indeed, if the current iteration's value for $\{f_X^{\mathbf{\Lambda}\dagger}\}$ is not unique, it is possible that the additional cone being added does not eliminate other solutions $\boldsymbol{v}$ that achieve the same value.

A few remarks are in order. It is evident that provided the initial step results in a feasible solution for $f_X^{\mathbf{\Lambda}\dagger}$, all other iterations will be feasible as we add further dual $\lambda$-witnessed AGE cone constraints. Therefore, one essential implementation task revolves around setting the initial $\Lambda_{init}$, ensuring $f_X^{\mathbf{\Lambda}^\dagger}$ results in a feasible solution initially and future iterations. Having tried numerous initialization procedures, our final initialization procedure for our numerical experiments was a uniform initialization. Specifically, we can initialize $\mathsf{\Lambda}$ to contain $m$ vectors, $\boldsymbol{\lambda}_i \in \mathsf{N}_i$ for each $i \in [m]$, where $\lambda_{ij} = 1/(m-1)$ for all $j \neq i$. More efficiently, if we are trying to produce the SAGE bound for a signomial $f = \mathrm{Sig}(\boldsymbol{A}, \boldsymbol{c})$, then we could choose to initialize $\mathsf{\Lambda}$ to include only $|\{i : c_i < 0, i \in [m]\}|$ vectors, one for each negative term of $f$.

In Algorithm 1, we start with an initialization procedure for $\boldsymbol{\lambda}$ and add $\boldsymbol{\lambda}$ according to the separation problem with 1 additional $\boldsymbol{\lambda}$ per iteration. To optimize the column generation and approach the SAGE bound faster, for the first few initial steps, more $\boldsymbol{\lambda}$ could be added into our set $\Lambda$ from the separation problem provided all $\boldsymbol{\lambda}$ added had a negative value in the separation problem. This could allow the first few steps to add more cones of interest in the generation. As more constraints are added into the generation, certain constraints previously generated in $\Lambda$ may not be binding as more recent constraints are binding. Since we are working in the dual space of the $\boldsymbol{\lambda}$-witnessed AGE cones, our feasible set for $f_X^{\mathbf{\Lambda}\dagger}$ is convex and any nonbinding constraint can be removed from the set of constraints while not affecting the optimization problem. This realization provides a way to remove extraneous $\boldsymbol{\lambda}$ from $\Lambda$ in later iterations, resulting in a easier problem while maintaining a relevant set $\Lambda$.

### 3.2 Relation to X-circuits

Finding X-circuits for a generic domain $X$ is generally a difficult problem. The following theorem provides an interesting connection between the constraint generation algorithm and the X-circuits for the domain of interest.

**Theorem 4.** *Suppose that X is an bounded set. Consider the minimization separation problem:*

$$\min_{\boldsymbol{\lambda}} \left\{ \boldsymbol{y}^\intercal \boldsymbol{\lambda} + \sigma_X(-\boldsymbol{A}^\intercal \boldsymbol{\lambda}) \ : \ \sum_{j \neq i} \lambda_j = 1, \lambda_i = -1, \lambda_j \geq 0 \, \textit{for all } i \neq j \right\} \tag{10}$$

*If there is a unique solution $\lambda^*$ to* (10)*, then $\lambda^*$ is an X-circuit.*

*Proof.* We will prove this theorem in three steps. First, we will transform the minimization problem in terms of a new domain $X_0$ which contains the origin and $\lambda^*$ is a unique solution to this minimization problem. Second, we will prove that $\lambda^*$ is a $X_0$-circuit of this shifted domain $X_0$. Finally, we will use the invariability of sublinear circuits under transformation to show that $\lambda^*$ must be a X-circuit.

For the first step, consider any point in $\boldsymbol{x}_0 \in X$. Let $X^*$ be the domain $\{\boldsymbol{x} - \boldsymbol{x}_0 : \boldsymbol{x} \in X\}$. Using the definition of the support function $\sigma_X(\boldsymbol{y})$, we have that:

$$\begin{aligned}
\boldsymbol{y}^\intercal \boldsymbol{\lambda^*} + \sigma_X(-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*}) &= \boldsymbol{y}^\intercal \boldsymbol{\lambda^*} + \sup\{(-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*})^\intercal \boldsymbol{x} : \boldsymbol{x} \in X\} \\
&= \boldsymbol{y}^\intercal \boldsymbol{\lambda^*} + \sup\{(\boldsymbol{x} + \boldsymbol{x}_0)^\intercal (-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*}) : \boldsymbol{x} \in X^*\} \\
&= \boldsymbol{y}^\intercal \boldsymbol{\lambda^*} + (-\boldsymbol{A}\boldsymbol{x}_0)^\intercal \lambda^* + \sup\{(-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*})^\intercal \boldsymbol{x} : \boldsymbol{x} \in X^*\} \\
&= (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_0)^\intercal \boldsymbol{\lambda^*} + \sigma_{X^*}(-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*})
\end{aligned}$$

So, we can say that $\boldsymbol{\lambda}^*$ is the unique solution to the problem $\min_\lambda\{\boldsymbol{y_1}^\intercal \boldsymbol{\lambda^*} + \sigma_{X^*}(-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*})\}$ where $y_1 = y - \boldsymbol{A}\boldsymbol{x}_0$.

For the second step, we know that $X^* = X - x_0$ is a bounded set which contains the origin. This implies that for a given $\boldsymbol{\lambda}$, we have that $0 \leq \sigma_X(-\boldsymbol{A}^\dagger \boldsymbol{\lambda}) < \infty$ since $\sigma_X(0)$ provides the lower bound and the boundedness of $X^*$ provides an upper bound.
Suppose for the sake of contradiction that $\boldsymbol{\lambda}^*$ is not an X-circuit. Since $\boldsymbol{\lambda}^* \in \overline{\mathsf{N}}_i \to \lambda_i \neq 0$ and $X$ is bounded implies $\sigma_X(-\boldsymbol{A}^\dagger \lambda^*) < \infty$, the only way for this to not be an X-circuit by definition is if $\boldsymbol{\lambda}^*$ can be written as a convex combination of $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in \overline{\mathsf{N}}_i$ for which $\boldsymbol{\lambda} \to \sigma_X(-\boldsymbol{A}^\intercal \lambda)$ is linear on $[\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2]$. This means that we can write $\boldsymbol{\lambda}^* = t\boldsymbol{\lambda}_1 + (1-t)\boldsymbol{\lambda}_2 = (1-s)\boldsymbol{\lambda}_1 + s\boldsymbol{\lambda}_2$ for $t, s \in (0, 1)$ such that linearity hold for the support function. Now, moving to the optimization problem in $X^*$, this implies that

$$\boldsymbol{y_1}^T(t\boldsymbol{\lambda}_1 + (1-t)\boldsymbol{\lambda}_2) + \sigma_{X^*}(-\boldsymbol{A}^\dagger(t\lambda_1 + (1-t)\lambda_2))$$

is unique minimal solution to $\min_\lambda\{\boldsymbol{y_1}^\intercal \boldsymbol{\lambda^*} + \sigma_{X^*}(-\boldsymbol{A}^\intercal \boldsymbol{\lambda^*})\}$.

Now, suppose that $\boldsymbol{y}^T(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2) \geq 0$. Applying the linearity condition, this can be rewritten as:

$$\boldsymbol{y}_1^T\boldsymbol{\lambda_2} + t\boldsymbol{y}_1^T(\boldsymbol{\lambda_1} - \boldsymbol{\lambda_2}) + \sigma_{X^*}(-\boldsymbol{A}^\intercal \boldsymbol{\lambda}_2) + t\sigma_{X^*}(-\boldsymbol{A}^\intercal(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)) \geq \boldsymbol{y}_1^T\boldsymbol{\lambda}_2 + \sigma_{X^*}(-\boldsymbol{A}^\dagger \boldsymbol{\lambda}_2)$$

where the last inequality follows since $\boldsymbol{y}^T(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2) \geq 0$ and $\sigma_{X^*}(-\boldsymbol{A}^\dagger(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)) \geq 0$.

The second case follows similar logic. In either case, we have that there exists another solution $\boldsymbol{\lambda}_i$ such that $\boldsymbol{\lambda}_i \in \overline{\mathsf{N}}_i$ which results in a lower objective value to the minimization problem than $\boldsymbol{\lambda}^*$. This is a contradiction implying that $\boldsymbol{\lambda}_*$ must have be an X-circuit.

For the final step, we know that the property of $\boldsymbol{\lambda}^*$ being an $X$-circuit is invariant under the translation of X. Thus, we can translate our set $X - x_0$ back to $X$ and $\lambda$ will still be an X-circuit, completing our proof. □

This theorem raises some interesting possibilities in finding sublinear circuits for a domain $X$. When $X$ is polyhedral, the minimization problem (10) reduces to a linear programming problem. Checking whether an optimal solution is unique for a linear program can be done with a transformation to another linear program described by Appa [18]. Thus, Theorem 4 raising an interesting line of work to find sublinear circuits for polyhedral circuits using a generative procedure where previous theoretical progress has been limited to simple polyhedral cases like the nonnegative orthant and cube $[-1, 1]^n$ [19].

### 3.3 Extension to $(p, q, l)$-hierarchy SAGE relaxation

In this section, we consider the $(p, q, l)$ hierarchy for SAGE relaxations and apply constraint generation to this extension. To extend our framework from Section 3.1 to the $(p, q, l)$ hierarchy for signomial relaxations, we define the moment reduction array function which is used in our definition of the dual. For a signomial $f = \text{Sig}(\boldsymbol{A}, c)$, let $f.\boldsymbol{A}$ and $f.\boldsymbol{c}$ be the parameters $\boldsymbol{A}$ and $c$ respectively. Suppose that we have a constraint $h$ which has been incorporated into a Lagrangian $L$ with an associated Lagrangian multiplier $s_h$. Suppose, $\boldsymbol{v}$ is a dual variable to the constraints $L \in \mathsf{C}_{\mathsf{SAGE}}(L.\boldsymbol{A}, X)$. If $s_h$ is constrained to be a SAGE function, the dual problem will contain a constraint of the form $\boldsymbol{C}_h^\intercal \boldsymbol{v} \in \mathsf{C}_{\mathsf{SAGE}}(s_h.\boldsymbol{A}, X)^\dagger$ where $\boldsymbol{C}_h$ is the output of the moment relaxation function. If $s_h$ is unconstrained, then the dual problem will include the constraint $\boldsymbol{C}_h^\intercal \boldsymbol{v} = 0$. Looking at the primal problem in Eq. 5 along with the moment relaxation function, we can represent the dual form of the hierarchy as:

$$(f,g,\phi)_X^{(p,q,l)\dagger} = \sup (f.\boldsymbol{c})^\intercal \boldsymbol{v} \text{ such that } L = f - \gamma - \sum_{h\in g[q]} s_h \cdot h - \sum_{h \in \phi[q]} z_h \cdot h$$
$$v \in \mathsf{C}_{\mathsf{SAGE}}(Sig(\boldsymbol{A},1)^l L.alpha, X)^\dagger$$
$$\boldsymbol{C}_{h1}^\intercal \boldsymbol{v} \in \mathsf{C}_{\mathsf{SAGE}}(s_h.\boldsymbol{A}, X)^\dagger \ \forall h \in g[q]$$
$$\boldsymbol{C}_{h2}^\intercal \boldsymbol{v} = 0 \ \forall h \in \phi[q]$$
$$\mathbf{1}^\intercal \boldsymbol{v} = 1 \tag{11}$$

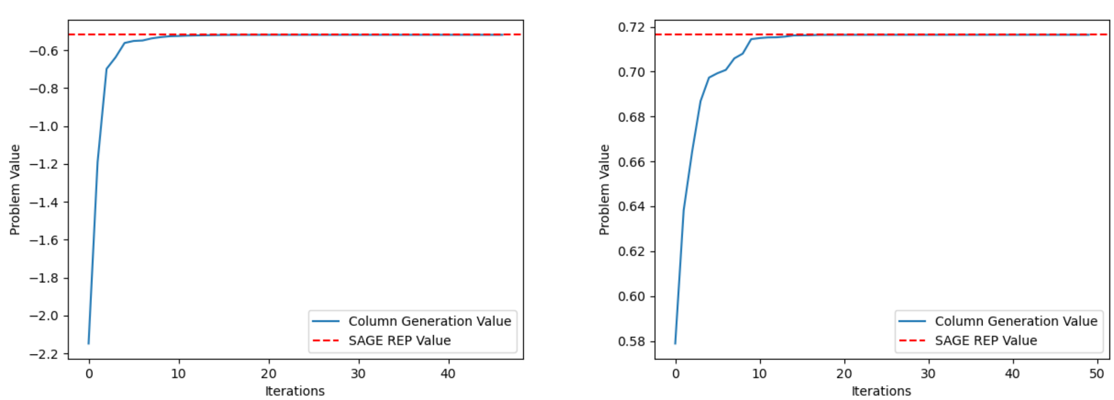
where $\boldsymbol{C}_{h1}$ and $\boldsymbol{C}_{h2}$ are the outputs of the moment relaxation function on $h \in g[q]$ and $h \in \phi[q]$ respectively. Extending our constraint generation to this dual form requires a simple variation to the dual form. Specifically, for every dual SAGE cone $\mathsf{C}_{\mathsf{SAGE}}(\boldsymbol{A}, X)$ constraint present in 11, we can replace dual cone with an outer approximation $\mathsf{C}_{\boldsymbol{\lambda}}(\boldsymbol{A}, X)^\dagger$. We can then perform Algorithm 1 on each individual set $\Lambda$, solving the optimization problem $(f,g,\phi)_X^{(p,q,l)\dagger}$ and getting values for associated variables $\boldsymbol{v}, \boldsymbol{C}_{h1}, \boldsymbol{C}_{h2}$ and updating each approximation according the separation problem in Algorithm 1. Examples of the use of this relaxation are presented in Section 4.2

# 4 Numerical Results

We present the results of computational experiments with constraint generation. In section 4.1, we focus on the (0, 1, 0) hierarchy case from Section 3.1 and study the convergence rates of the constraint generation algorithm. Additionally, we perform a sensitivity analysis on the dependence of the number of variables and terms on the convergence rate. In section 4.2, we look at examples of signomial programs in literature previously solved using SAGE relaxations with the relative entropy approach. We apply our constraint generation to this problem and compare the two approaches. All experiments were done on a Lenovo Legion Y740 Laptop with 16GB of RAM using the MOSEK solver with inputted parameters of {"MSK_DPAR_INTPNT_CO_TOL_NEAR_REL": 1000}. Implementations were done using the SAGEOPT python package.

## 4.1 Randomized Experiments

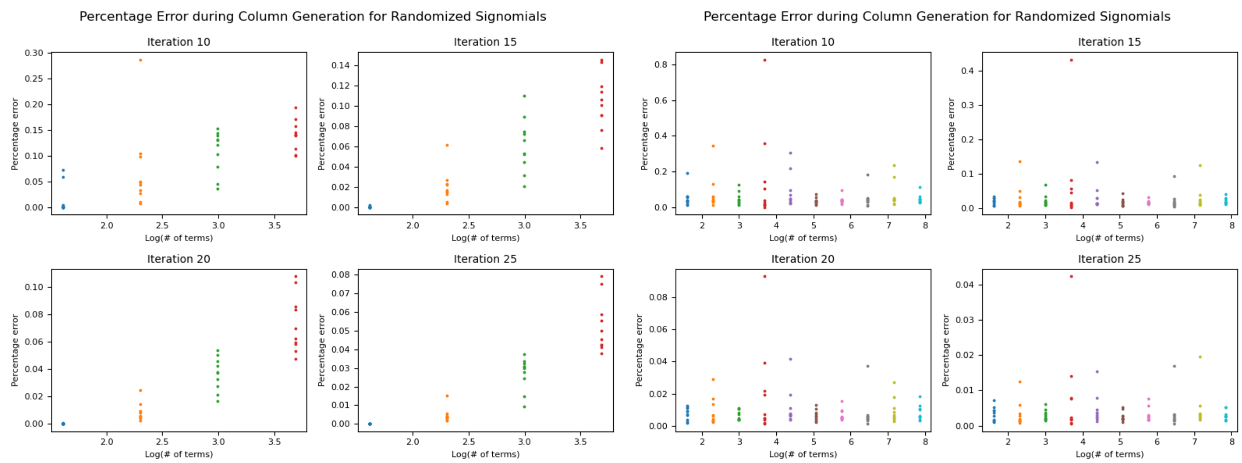
We ran the constraint generation algorithm for various randomized signomials optimized over the unit ball $|x|_2 \leq 1$ for $x \in \mathbb{R}^n$. We generated the signomials by randomly drawing coefficients $c$ and matrix $\boldsymbol{A}$ from a standard normal distribution. The matrix $\boldsymbol{A}$ was then normalized with a scaling factor to ensure $\sigma_X(-A^\dagger \lambda)$ was bounded for all randomized signomials. This refactor helped avoid large minimization values which could be overflow in the initial step of Algorithm 1. Figure 1 shows two examples of the column generation approach for randomly generated signomials. These signomials were optimized using the SAGE REP program along with the column generation whose value was plotted at each iteration.




(a) 4 term, 6 variable random signomial (b) 7 term, 20 variable random signomial

Figure 1: Convergence of column generation approach to SAGE bound over 50 iterations

One noticeable feature of the column generation when testing with randomized signomials was the near exponential convergence to the SAGE bound. We further analyzed these convergence rates. Figure 2 displays the percentage error of the column generation approach at different iterations for 10 randomized signomial problems. We observed that the number of variables in the signomial does not have a noticeable effect on the convergence rates of this approach while varying the terms has an increase in the convergence rates as the number of terms increase. This is promising since it gives evidence that this approach can reach a designated error of the SAGE bound independent of the number of variables in the SAGE bound, where SAGE can begin to struggle. For instance, if a signomial has $\sim 1000$ variables, SAGE's original relative entropy approach struggles to find the exact SAGE bound, However, reaching within 1% of the solution can be done with 20 iterations of the column generation with relative ease. In terms of varying the number of terms in the signomial, the convergence rates still similar exponential convergence rates between the varying number of terms. However, the initial percentage error was seen to be higher with increasing terms which resulted in larger terms signomials taking longer to reach the same error percentage of other iterations.




(a) Varying number of terms in signomial

(b) Varying number of variables in signomial

Figure 2: Percentage Convergence to SAGE bound per set iterations as the number of terms and variables increased

We also looked at the convergence rates of the column generation as the number of variables increased and observed near exponential convergence demonstrated by the semilog plot in Figure 3. A further avenue of research could be in justifying and properly quantifying this exponential convergence.

### 4.2 Problems in Literature

Here, we describe our comparisons between the SAGE REP and the constraint generation problem for 3 separate signomial problems taken from the 2014 article by Hou, Shen and Chen [20]. Summary results can be seen in Table 1. There are some interesting observations to take from the table. First, the (0, 3, 0)-hierarchy of Problem 2 is an example of a case where the SAGE relative entropy programming approach displays numerical issues; what should have resulted in a lower bound for the minimization problem gives us a value which is grossly above the known bound. However, the column generation approach value of the SAGE bound does not exhibit this issue with a bound of 11.952 being well below the known bound. Interestingly, in 25 iterations the column generation completely terminated. This implies that our column generation approach generated $\lambda$ which were able to search the *entire* SAGE cone as opposed to giving us a bound from a subset of the SAGE cone.

| Problem | Known Bound for Problem | (p, q, l) | SAGE REP Bound | Column Generation | Iterations Run |
|---:|---:|---|---:|---:|---:|
| **1** | 1.765082089 | (0, 1, 0) | 1.765082089 | 1.765082089 | 25 |
| **2** | 11.964 | (0, 3, 0) | 14.9935678 | 11.95203794 | 25 |
| | | (3, 3, 0) | 11.964 | Memory Error | |
| **3** | -147.6591 | (0, 2, 1) | -147.6747034 | -147.6783575 | 75 |

Table 1: A table of results when applying our column generation approach for the first 3 problems in the 2014 article by Hou, Shen and Chen. SAGE bounds with REP and column generation along with the known bound are recorded.

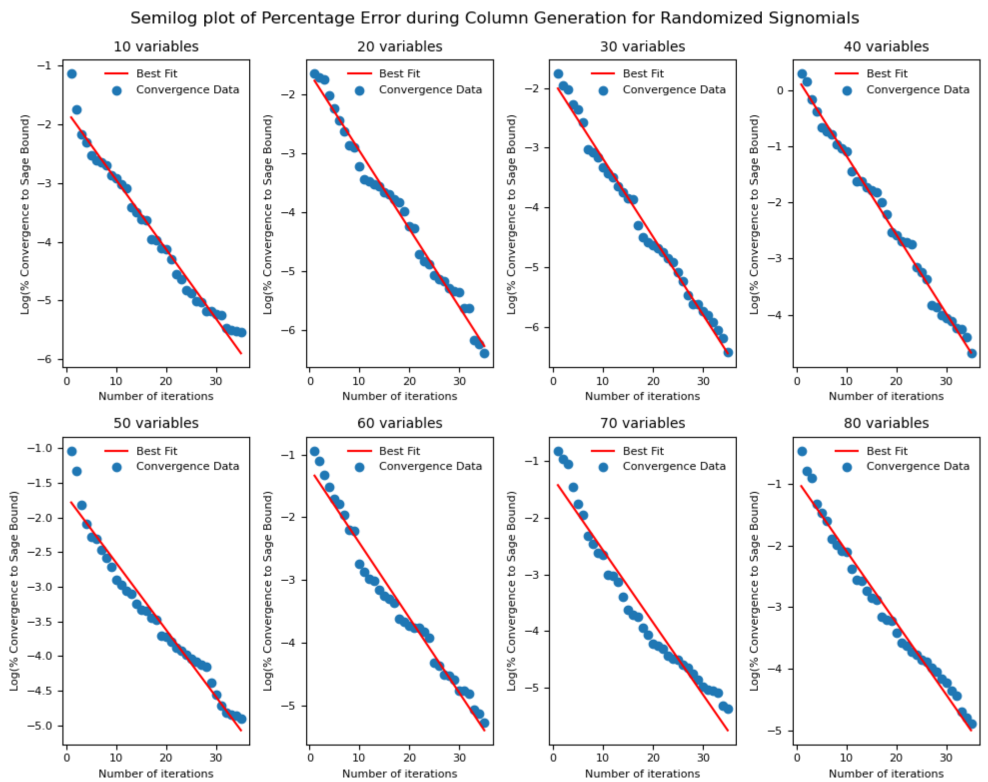



Figure 3: Semilog plot of convergence rates for Column Generation with uniform initial lambda

## 5 Conclusion and Outlooks

We extended SAGE decompositions further using a connection between SAGE certificate cones and the more elementary power cone. For compact $X \subset \mathbb{R}^n$, we developed an algorithm which iteratively generates power cones to approximate our SAGE cone. We also find a connection between this algorithm and certain properties of the domain of interest under mild conditions. The algorithm is also extended to fit previous hierarchies for signomial programming. Having tested our algorithm on randomly generated problems, we observed an exponential rate of convergence to the SAGE bound. Testing our algorithm on problems in previous literature, we saw cases where the relative entropy approach displayed numerical issues while the column generation correctly search the SAGE cone. These values

There are varying lines of future research from this work. Firstly, we aim to develop a more rigorous argument about the observed exponential convergence. Secondly, Theorem 3 can be implemented to give a generative algorithm to find and recognize X-circuits. Such implementations could be very useful theoretically as previous work to find X-circuits has posed many challenges. Further, finding such X-circuits could act as a presolve to our column generation algorithm. Using such X-circuits as part of the initial $\lambda$ could ensure that we reach regions of the SAGE bound which were not initially found with our uniform initialization procedure. Finally, on the same vein, there is significant room for improvement on our initialization procedures of our algorithm. Though the uniform initialization procedure worked the best out of the methods attempted, it can be modified or overhauled entirely for a more sophisticated procedure.

## References

[1] Martin A. York, Warren W. Hoburg, and Mark Drela. Turbofan engine sizing and tradeoff analysis via signomial programming. *Journal of Aircraft*, 55(3):988–1003, May 2018. doi:10.2514/1.c034463. URL `https://doi.org/10.2514/1.c034463`.

[2] Martin A. York, Berk Öztürk, Edward Burnell, and Warren W. Hoburg. Efficient aircraft multidisciplinary design optimization and sensitivity analysis via signomial programming. *AIAA Journal*, 56(11):4546–4561, November 2018. doi:10.2514/1.j057020. URL `https://doi.org/10.2514/1.j057020`.

[3] Berk Ozturk and Ali Saab. Optimal aircraft design decisions under uncertainty via robust signomial programming. In *AIAA Aviation 2019 Forum*. American Institute of Aeronautics and Astronautics, June 2019. doi:10.2514/6.2019-3351. URL `https://doi.org/10.2514/6.2019-3351`.

[4] Philippe G. Kirschen and Warren W. Hoburg. The power of log transformation: A comparison of geometric and signomial programming with general nonlinear programming techniques for aircraft design optimization. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, January 2018. doi:10.2514/6.2018-0655. URL `https://doi.org/10.2514/6.2018-0655`.

[5] Philippe G. Kirschen, Martin A. York, Berk Ozturk, and Warren W. Hoburg. Application of signomial programming to aircraft design. *Journal of Aircraft*, 55(3):965–987, May 2018. doi:10.2514/1.c034378. URL `https://doi.org/10.2514/1.c034378`.

[6] R.A. Jabr. Inductor design using signomial programming. *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, 26(2):461–475, 2007.

[7] Mung Chiang. *Nonconvex Optimization for Communication Networks*, pages 137–196. Springer US, Boston, MA, 2009. ISBN 978-0-387-75714-8.

[8] K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[9] Venkat Chandrasekaran and Parikshit Shah. Relative entropy relaxations for signomial optimization. *SIAM Journal on Optimization*, 26(2):1147 – 1173, 2016. doi:10.1137/140988978.

[10] Riley Murray, Venkat Chandrasekaran, and Adam Wierman. Newton polytopes and relative entropy optimization. *Foundations of Computational Mathematics*, 03 2021. doi:10.1007/s10208-021-09497-w.

[11] Riley Murray, Venkat Chandrasekaran, and Adam Wierman. Signomial and polynomial optimization via relative entropy and partial dualization. *Mathematical Programming Computation*, 13, 10 2020. doi:10.1007/s12532-020-00193-4.

[12] Qing Ye and Weijun Xie. Second-order conic and polyhedral approximations of the exponential cone: Application to mixed-integer exponential conic programs, 2021.

[13] Scott B. Lindstrom, Bruno F. Lourenço, and Ting Kei Pong. Error bounds, facial residual functions and applications to the exponential cone, 2020.

[14] MOSEK ApS. *MOSEK Modeling Cookbook*, 2021.

[15] Robert Chares. Cones and interior-point algorithms for structured convex optimization involving powers andexponentials, 2009.

[16] R. Murray. *Applications of convex analysis to signomial and polynomial nonnegativity problems*. PhD thesis, California Institute of Technology, 6 2021.

[17] Dávid Papp. Duality of sum of nonnegative circuit polynomials and optimal sonc bounds, 2019.

[18] G. Appa. On the uniqueness of solutions to linear programs. *The Journal of the Operational Research Society*, 53 (10):1127–1132, 2002. ISSN 01605682, 14769360. URL `http://www.jstor.org/stable/822972`.

[19] Helen Naumann and Thorsten Theobald. Sublinear circuits for polyhedral sets, 2021.

[20] Xue-Ping Hou, Peiping Shen, and Yong-Qiang Chen. A global optimization algorithm for signomial geometric programming problem. *Abstract and Applied Analysis*, 2014:1–12, 03 2014. doi:10.1155/2014/163263.