Compute-and-Forward: Finding the Best Equation

Saeid Sahraei and Michael Gastpar[1]

***Abstract*— Compute-and-Forward is an emerging technique to deal with interference. It allows the receiver to decode a suitably chosen integer linear combination of the transmitted messages. The integer coefficients should be adapted to the channel fading state. Optimizing these coefficients is a Shortest Lattice Vector (SLV) problem. In general, the SLV problem is known to be prohibitively complex. In this paper, we show that the particular SLV instance resulting from the Compute-and-Forward problem can be solved in low polynomial complexity and give an explicit deterministic algorithm that is guaranteed to find the optimal solution.**

I. INTRODUCTION

Compute-and-Forward is an emerging paradigm for dealing with interference in wireless multiuser channels. Contrary to previous approaches, Compute-and-Forward does not view interference as inherently undesirable. The key idea is that the relay nodes aim at recovering integer linear combinations of transmitted codewords instead of attempting to decode the individual codewords and treating interference as noise. Nested lattice codes ensure that an integer linear combination of codewords is a codeword itself. This method is shown to considerably enhance the achievable rates compared to other techniques [1], [2], [3].

In a typical Compute-and-Forward scenario, n transmitting nodes, each with transmission power P, share a wireless channel to send their messages to a relay node. The relay receives a noisy linear combination of the transmitted messages, namely

$$y=\sum_{i=1}^{n} h_i x_i + z$$

where y is the received signal, x_i and h_i respectively represent the signal transmitted by node i and the effect of channel from this node to the relay and z denotes additive white Gaussian noise of unit variance. The receiver then recovers an integer linear combination of the transmitted codewords (Figure 1). It is proved in [1] that the achievable rate for the Compute-and-Forward scheme is at least as large as:

$$R(\mathbf{h},\mathbf{a}) = \frac{1}{2}\log^{+}\left(\left(\|\mathbf{a}\|^2 - \frac{P|\mathbf{h}^T\mathbf{a}|^2}{1+P\|\mathbf{h}\|^2}\right)^{-1}\right) \tag{1}$$

where $\mathbf{a}$ describes the coefficient vector of the linear combination of messages to be recovered at the relay node.

From the perspective of a single receiver, a reasonable choice for the vector $\mathbf{a}$ is one that maximizes the achievable rate given by (1), that is:

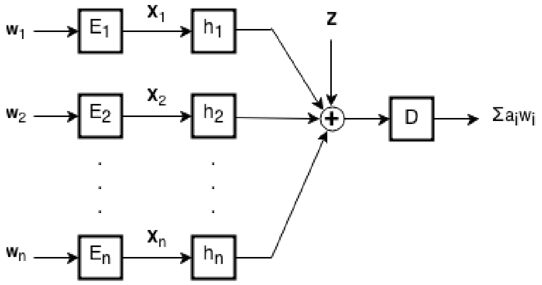

Fig. 1. n transmitters send their signals to one relay. The relay decodes an integer linear combination of the codewords.

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathbb{Z}^n\setminus\{\mathbf{0}\}}{\arg\max}\ \frac{1}{2}\log^{+}\left(\left(\|\mathbf{a}\|^2 - \frac{P|\mathbf{h}^T\mathbf{a}|^2}{1+P\|\mathbf{h}\|^2}\right)^{-1}\right). \tag{2}$$

It is shown in [1] that $\mathbf{a}^*$ satisfies

$$\|\mathbf{a}^*\| \le \sqrt{P\|\mathbf{h}\|^2+1}. \tag{3}$$

Given the bounded interval, one can attempt to solve problem (3) by exhaustive search. For a small number of users n this is feasible, but it quickly becomes prohibitively complex as n grows large.

With slight manipulation, this optimization problem can be rewritten as

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathbb{Z}^n\setminus\{\mathbf{0}\}}{\arg\min}\ f(\mathbf{a}) = \mathbf{a}^T\mathbf{G}\mathbf{a} \tag{4}$$

where

$$\mathbf{G} = (1+P\|\mathbf{h}\|^2)\mathbf{I} - P\mathbf{h}\mathbf{h}^T.$$

is a positive definite matrix. This is an instance of a well-known problem in discrete mathematics called the Shortest Lattice Vector problem. The SLV problem is known to be NP-hard in its general form, that is for a general positive definite matrix $\mathbf{G}$. Due to this fact, it has been suggested that approximation algorithms must be used in order to solve (4). The problem with such algorithms is that the best known polynomial complexity approximation algorithms for the SLV problem have exponential approximation factors. The most famous among them are the celebrated LLL algorithm [4] and its extensions, most notably [5]. In fact, Khot in [6] has shown that assuming $NP \not\subseteq RP$, no constant factor approximation algorithm can be found for the SLV problem which runs in polynomial complexity. Other results on hardness of the SLV problem have been found, for instance in [7], [8], [9].

[1] Both authors are with the Department of Computer and Communication Sciences, EPFL, Switzerland `saeid.sahraei@epfl.ch` , `michael.gastpar@epfl.ch`

On the bright side, efficient algorithms for special lattices have been known for a long time. For instance Gauss found an algorithm for solving the SLV problem in dimension two. Conway in [10] provides exact algorithms for a class of root lattices in higher dimensions. Based on [11] McKilliam [12] showed that if an obtuse superbase for a lattice is known, the shortest vector can be found in polynomial complexity.

Since the publication of Compute-and-Forward paper [1], there has been a few attempts to tackle the special case of SLV problem which appears in (4). Performance of the proposed methods are mostly exhibited through simulation results or heuristic arguments and NP-hardness of the problem in hand is typically the underlying assumption [13], [14], [15].

In this work we show that the special case of SLV problem which appears in Compute-and-Forward admits an exact solution of polynomial complexity. The following theorem, albeit provable mostly by elementary manipulations of integer inequalities, establishes an important fact that provides the foundation of our SLV algorithm. (Note that the operator $\lceil\cdot\rfloor$ rounds each element of its vector input to the closest integer.)

Theorem 1. *The solution to (4) satisfies*

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{h}x < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$$

and thus

$$\mathbf{a}^* = \lceil \mathbf{h}x \rfloor$$

for some $x \in \mathbf{R}$. Or $\mathbf{a}^*$ must be a standard unit vector, up to a sign.

It follows from Theorem 1 that for the special lattices of interest, the shortest vector can be obtained by solving an optimization problem over only one variable. Equation (3) tells us the search only has to be done over a bounded region. An individual examination of the standard unit vectors must also be performed. This will significantly reduce the number of candidate vectors $\mathbf{a}$. We will find an upper bound on this number, propose a method to enumerate all such candidates and find the one that minimizes f as defined in (4). We will prove that the complexity of our algorithm is

$$O\left(n^2\sqrt{1+P\|\mathbf{h}\|^2}\right).$$

Remark 1. The formula given by Theorem 1 has some resemblance to the results of [16] and [17]. However the span of these works are Coxeter lattices and the goal is to find faster algorithms for problems which are already known to be polynomially solvable.

The rest of the paper is organized as follows: First we define the notation used throughout the paper. In Section III we introduce an algorithm which solves the optimization problem (4) based on Theorem 1. The complexity of the algorithm will then be calculated. This will be followed by a generalization of Theorem 1 and the corresponding algorithm to the case of MIMO Compute-and-Forward respectively in Sections IV and V. An analysis of the complexity of this algorithm will be given in Section V-A. In Section VI the proof of Theorem 2 which is a generalized version of Theorem 1 is given. Finally we will conclude our work in Section VII.

II. Notation

We use boldface lowercase letters to denote vectors. All vectors are assumed to be vertical. In particular we use $\mathbf{1}$ to denote the all-ones vector and $\mathbf{0}$ for the all-zero vector. Boldface capital letters represent matrices. Scalars are written with plain letters. For example, for a matrix $\mathbf{A}$ we use A_{ij} to refer to the element in its i-th row and j-th column. Similarly, for the vector $\mathbf{a}$, we denote its i-th element by a_i. When referring to indexed vectors, we use boldface letters. For instance, $\mathbf{a}_i$ denotes the i-th vectors, whereas a_{ij} indicates the j-th element of the i-th vector.
For an $n \times m$ matrix $\mathbf{A}$ and for a set $\pi \subseteq \{1,\ldots,n\}$ we define $\mathbf{A}_\pi$ as the submatrix of $\mathbf{A}$ which consists of the rows indexed in π. For a vector $\mathbf{a}$, we define $\mathbf{a}_\pi$ in a similar manner. For an $n \times n$ matrix $\mathbf{A}$ we use $diag(\mathbf{A})$ to denote a vector consisting of its diagonal elements.
All the vector inequalities used throughout the paper are elementwise. The operator $\lceil\cdot\rceil$ returns the smallest integer greater or equal to its input. The two operators $\lceil\cdot\rfloor$ and $\lfloor\cdot\rceil$ return the closest integer to their input. Their difference is at half-integers: the former rounds the half-integers up and the latter rounds them down. We use $\|\cdot\|$ to represent the 2-norm of a vector. Finally, $\mathbb{R}$ represents the set of real numbers and $\mathbb{Z}$ the set of integers.

III. Algorithm I

In this section we provide an efficient algorithm for solving the optimization problem (4). We will show that the algorithm runs in polynomial complexity in n. For the sake of convenience we define $\psi = \sqrt{1+P\|\mathbf{h}\|^2}$.

In line with Theorem 1 define $\mathbf{a}(x) = \lceil \mathbf{h}x \rfloor$. Note that Theorem 1 reduces the problem to a one-dimensional optimization task. Since every $a_i(x)$ is a piecewise constant function of x, so is the objective function f. Overall, the goal is to find a set of points which fully represent all the intervals in which f is constant and choose the point that minimizes f. Being a piecewise constant function, f can be represented as:

$$f(\mathbf{a}(x)) = \begin{cases} f_i \ , & \text{if } \xi_i < x < \xi_{i+1} \ , \ i = \ldots,-1,0,1,\ldots \\ h_i \ , & \text{if } x = \xi_i \ , \ i = \ldots,-1,0,1,\ldots \end{cases} \tag{5}$$

ξ_i values are sorted real numbers denoting the points of discontinuity of f. Since f is a continuous function of $\mathbf{a}$, these are in fact the discontinuity points of $\mathbf{a}(x)$ (or a subset of them) or equivalently the points where $a_i(x)$ is discontinuous, for some $i = 1\ldots n$. Since we have that $a_i(x) = \lceil h_i x \rfloor$, the discontinuity points of $a_i(x)$ are the points where $h_i x$ is a half-integer. Or equivalently the points of the form $x = \frac{c}{|h_i|}$ where c is a half-integer and $h_i \neq 0$. To conclude this argument, we write:

$$\xi_i \in \left\{ \frac{c}{|h_j|} \;\middle|\; j = 1 \dots n \,, h_j \neq 0 \,,\, c - \frac{1}{2} \in \mathbb{Z} \right\} \qquad (6)$$
$$i = \dots, -1, 0, 1, \dots$$

We can also see from Theorem 1 that $\mathbf{a}^*$ satisfies $\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{h}x < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$ for some $x \in \mathbb{R}$. Hence any x satisfying

$$a_i^* - \frac{1}{2} < xh_i < a_i^* + \frac{1}{2} \;,\; i = 1 \dots n \,,\, h_i \neq 0 \qquad (7)$$

minimizes f. As a result, x belongs to the interior of an interval and not the boundary. Therefore, in the process of minimizing f, one can ignore the h_i values in (5), check all f_i values and choose the smallest one.

$$\min_{\mathbf{a} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}} f(\mathbf{a}) = \min_{i = \cdots -1, 0, 1 \dots} f_i$$

Since $\frac{\xi_i + \xi_{i+1}}{2}$ belongs to the interval (ξ_i, ξ_{i+1}), we can rewrite f_i as $f_i = f(\mathbf{a}(\frac{\xi_i + \xi_{i+1}}{2}))$. Thus:

$$\min_{\mathbf{a} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}} f(\mathbf{a}) = \min_{i = \cdots -1, 0, 1 \dots} f(\mathbf{a}(\frac{\xi_i + \xi_{i+1}}{2})) \qquad (8)$$

On the other hand, (3) tells us that we do not need to check the whole range of x. It follows from the constraint $\|\mathbf{a}\| \leq \psi = \sqrt{1 + P\|\mathbf{h}\|^2}$ that $|a_i| \leq \psi$ and thus, from (7) we have

$$-\frac{1}{|h_i|}(\psi + \frac{1}{2}) < x < \frac{1}{|h_i|}(\psi + \frac{1}{2}) \;,\; i = 1 \dots n \,,\, h_i \neq 0$$
$$\Rightarrow -\beta(\psi + \frac{1}{2}) < x < \beta(\psi + \frac{1}{2})$$

where

$$\beta = \min_{i = 1 \dots n \,,\, h_i \neq 0} \frac{1}{|h_i|} \qquad (9)$$

It follows from this expression and equation (6) that the largest ξ_{i+1} that we need to check in equation (8) is $\beta(\lceil \psi \rceil + \frac{1}{2})$. Similarly the smallest ξ_i to be checked is $-\beta(\lceil \psi \rceil + \frac{1}{2})$. We can now rewrite equation (8) as:

$$\min_{\mathbf{a} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}} f(\mathbf{a}) = \min_{\substack{i = \cdots -1, 0, 1 \dots \\ \xi_i \geq -\beta(\lceil \psi \rceil + \frac{1}{2}) \\ \xi_{i+1} \leq \beta(\lceil \psi \rceil + \frac{1}{2})}} f(\mathbf{a}(\frac{\xi_i + \xi_{i+1}}{2})) \qquad (10)$$

Using equation (6) we can translate the constraints in equation (10) into:

$$\frac{c}{|h_j|} \leq \beta(\lceil \psi \rceil + \frac{1}{2}) \Rightarrow c \leq \lceil \psi \rceil + \frac{1}{2} \,,\, j = 1 \dots n \,, \text{ and} \qquad (11)$$

$$\frac{c}{|h_j|} \geq -\beta(\lceil \psi \rceil + \frac{1}{2}) \Rightarrow c \geq -(\lceil \psi \rceil + \frac{1}{2}) \,,\, j = 1 \dots n \qquad (12)$$

By defining the sets Φ_j , $j = 1 \dots n$ and the set Φ as follows:

$$\Phi_j = \left\{ \frac{c}{|h_j|} \;\middle|\; |c| \leq \lceil \psi \rceil + \frac{1}{2} \,,\, c - \frac{1}{2} \in \mathbb{Z} \right\} \qquad (13)$$
$$j = 1 \dots n \,,\, h_j \neq 0$$
$$\Phi_j = \emptyset \;,\; j = 1 \dots n \,,\, h_j = 0 \qquad (14)$$
$$\Phi = \bigcup_{j=1}^{n} \Phi_j \qquad (15)$$

and after sorting the elements of Φ, we can write the equation (10) as

$$\min_{\mathbf{a} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}} f(\mathbf{a}) = \min_{\xi_i \,,\, \xi_{i+1} \in \Phi} f(\mathbf{a}(\frac{\xi_i + \xi_{i+1}}{2})) \qquad (16)$$

Thus, the algorithm starts by calculating the sets Φ_j and their union Φ, sorting the elements of Φ and then running the optimization problem described by equation (16). The standard unit vectors will also be individually checked. The number of elements in Φ_j is upper-bounded by $2\lceil \psi \rceil + 2$ and thus the number of elements in Φ is upper-bounded by $n(2\lceil \psi \rceil + 2)$. Also the value of f can be calculated in $O(n)$. So the complexity of the algorithm is $O(n^2\sqrt{1 + P\|\mathbf{h}\|^2})$. The procedure is summarized in Algorithm 1.

Algorithm 1 Finding the optimal coefficient vector

```
Input: Channel vector $\mathbf{h}$ and transmission power P
Output: $\mathbf{a}^*$
Initialization :
$\mathbf{u}_i :=$ standard unit vector in the direction of $i$-th axis
$\psi := \sqrt{1 + P\|\mathbf{h}\|^2}$
$\Phi = \emptyset$
$\mathbf{G} := (1 + P\|\mathbf{h}\|^2)\mathbf{I} - P\mathbf{h}\mathbf{h}^T$
$f(\mathbf{a}) := \mathbf{a}^T\mathbf{G}\mathbf{a}$
$f_{\min} = \min(diag(\mathbf{G}))$
$\mathbf{a}^* = \mathbf{u}_{\arg\min(diag(\mathbf{G}))}$

Phase 1:
for all $i \in \{1, \dots, n\}$, and $h_i \neq 0$ do
    for all $c$ , $|c| \leq \lceil \psi \rceil + \frac{1}{2}$ and $c - \frac{1}{2} \in \mathbb{Z}$ do
        calculate $x = \frac{c}{|h_i|}$
        Set $\Phi = \Phi \cup \{x\}$
    end for
end for

Phase 2:
sort $\Phi$
for every two consecutive members of $\Phi$ do
    calculate $x$ = average of the two points
    calculate $\mathbf{a} = \lceil \mathbf{h}x \rfloor$
    if $f(\mathbf{a}) < f_{\min}$ AND $\mathbf{a}$ is not the all zero vector then
        set $\mathbf{a}^* = \mathbf{a}$
        set $f_{\min} = f(\mathbf{a})$
    end if
end for
return $\mathbf{a}^*$
```

IV. A GENERALIZATION TO MIMO COMPUTE-AND-FORWARD

In this section we provide a generalization of Theorem 1 and the corresponding algorithm by relaxing several constraints that we imposed on the structure of the Gram matrix. The generalized theorem can be applied to maximize the achievable rate of MIMO Compute-and-Forward studied in [18]. The scenario is very similar to what mentioned in the introduction, with the difference that the relay node now has multiple antennas. The objective remains the same: decode the best integer linear combination of the received codewords. Assume there are n transmitters with transmission power P and the receiver node has k antennas. Let $\mathbf{h}_i$ be the channel vector from the transmitting nodes to the i-th antenna of the relay. Also, let $\mathbf{H}$ be the $n \times k$ matrix whose columns are the $\mathbf{h}_i$ vectors. It directly follows from the results of [18] that the achievable rate satisfies the following equation:

$$R(\mathbf{a}) = -\frac{1}{2}\log \mathbf{a}^T \mathbf{G}\mathbf{a} \tag{17}$$

where

$$\mathbf{G} = \mathbf{W}\mathbf{R}\mathbf{W}^T. \tag{18}$$

Here $\mathbf{W}$ is a unitary matrix in $\mathbb{R}^{n\times n}$ whose columns are the eigenvectors of $\mathbf{H}\mathbf{H}^T$, and $\mathbf{R}$ is a diagonal square matrix with the first k diagonal elements satisfying

$$r_i = \frac{1}{1+P\gamma_i^2} \ , \ i = 1\cdots k$$

and the last $n-k$ diagonal elements equal to 1. Finally, γ_i^2 is the i-th eigenvalue of $\mathbf{H}\mathbf{H}^T$ (same order as the columns of $\mathbf{W}$).

The goal is to maximize the achievable rate or equivalently, to find

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathbb{Z}^n\setminus\{\mathbf{0}\}}{\arg\min}\, \mathbf{a}^T\mathbf{G}\mathbf{a}$$

as in the single antenna case.

We first mention a generalization of Theorem 1 and next we show that the Gram matrix which appears in this optimization problem satisfies the constrains of the new theorem. To begin with, we define the following:

Definition 1 (DP^k decomposable matrices)**.** We call a positive definite matrix DP^k decomposable if it can be written as $\mathbf{G} = \mathbf{D} - \mathbf{P}$ where $\mathbf{P}$ is a positive semi-definite matrix of rank k and $\mathbf{D}$ is a diagonal matrix. We call such a representation a DP^k decomposition of the matrix $\mathbf{G}$.

It follows from the definition that all diagonal elements of $\mathbf{D}$ are strictly positive. This is due to the fact that $\mathbf{G}$ is positive definite and $\mathbf{P}$ is positive semi-definite. Furthermore, we find it convenient to write $\mathbf{P}$ as $\mathbf{P} = \mathbf{V}\mathbf{V}^\mathbf{T}$ where $\mathbf{V}$ is an $n\times k$ matrix whose columns are linearly independent. Such a decomposition is not unique, but our arguments will be valid regardless of how the matrix $\mathbf{V}$ is chosen. We will use this notation throughout the paper without redefining them.

Theorem 2. *Assume a positive definite matrix* $\mathbf{G}$ *is* DP^k *decomposable, that is* $\mathbf{G} = \mathbf{D} - \mathbf{V}\mathbf{V}^\mathbf{T}$ *as defined. Let* $\mathbf{a}^*$ *be a solution to*

$$\underset{\mathbf{a}\in\mathbb{Z}^n\setminus\{0\}}{\arg\min}\, f(\mathbf{a}) = \mathbf{a}^\mathbf{T}\mathbf{G}\mathbf{a}$$

Then both the following statements are true:
a) *there exists a vector* $\mathbf{x}\in\mathbb{R}^k$ *such that* $\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D}^{-\mathbf{1}}\mathbf{V}\mathbf{x} < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$ *and thus* $\mathbf{a}^* = \lceil \mathbf{D}^{-\mathbf{1}}\mathbf{V}\mathbf{x} \rfloor$. *Or* $\mathbf{a}^*$ *must be a standard unit vector, up to a sign.*
b) $\|\mathbf{a}^*\| \leq \psi = \sqrt{\frac{G_{min}}{\lambda_{min}}}$ *where* G_{min} *is the smallest diagonal element of* $\mathbf{G}$ *and* λ_{min} *is the smallest eigenvalue of* $\mathbf{G}$.

Note that Theorem 1 is a special case of Theorem 2 where $k=1$, $\mathbf{D} = (1+P\|\mathbf{h}\|^2)\mathbf{I}$ and $\mathbf{P} = P\mathbf{h}\mathbf{h}^T$. The bound on the norm of $\mathbf{a}^*$ given by (3) is also equivalent to the bound given in part b) of Theorem 2 , since we have $\lambda_{min} = 1$ and $G_{min} \leq 1 + P\|\mathbf{h}\|^2$ in the lattice studied in Theorem 1.

The Gram matrix in equation (18) also satisfies the constraints of Theorem 2: Since $\mathbf{W}$ is a unitary matrix, $\mathbf{G}$ can be rewritten as $\mathbf{I} - \mathbf{W}(\mathbf{I}-\mathbf{R})\mathbf{W}^T$. Clearly, $\mathbf{W}(\mathbf{I}-\mathbf{R})\mathbf{W}^T$ is of rank k (since $\mathbf{I}-\mathbf{R}$ has only k non-zero diagonal entries), and positive semi-definite. The bound given by part b) of the theorem translates into $\|\mathbf{a}\| \leq \sqrt{1+P\gamma_{max}^2}$ where γ_{max} is the maximum γ_i value. This is because $G_{min} \leq 1$ and the eigenvalues of $\mathbf{G}$ are easily seen to be equal to $\frac{1}{1+P\gamma_i^2}$ (with the same eigenvectors as $\mathbf{H}\mathbf{H}^T$) or 1.

Remark 2. The SLV problem is easy to solve for diagonal Gram matrices: as the given basis is already orthogonal, the length of the shortest vector is the square root of the minimum diagonal element of the Gram matrix. Theorem 2 implies that subtracting a positive semi-definite low rank perturbation from a diagonal Gram matrix does not change the property of being polynomially solvable.

Remark 3. Deciding whether a matrix is DP^k decomposable or not is outside of the scope of this work. Throughout this paper we will assume that the DP^k decomposition of the Gram matrix is given a priori. The interested reader is referred to [19], [20] for the state of the art algorithms which, under a set of conditions, can find the DP^k decomposition of a matrix, with minimal k.

V. ALGORITHM II

For the case $k=1$ we presented an algorithm which finds precisely one point inside every interval in which f is constant. For the general case, it is not clear to us how to find exactly one point per region. As a result we will present an algorithm which finds multiple points per region, while guaranteeing that first, every region has at least one representative point, and second, the number of points remains manageable, in the sense that it grows only as a polynomial function of n.

From Theorem 2 we know that the vector $\mathbf{a}^*$ satisfies the

$2n$ inequalities:

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D^{-1}Vx} < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$$

for some $\mathbf{x}$. In other words, $\mathbf{x}$ belongs to the interior of the polytope described by these constraints. By analogy to the case $k = 1$, we start by finding the set of vertices of all such polytopes. Each vertex is the intersection of at least k linearly independent hyperplanes of the form $c_i = (\mathbf{D^{-1}V})_{\{i\}}\mathbf{x}$, for half-integer c_i. Thus in order to find a vertex, we choose any set $\pi \subseteq \{1,\ldots,n\}$ for which $|\pi| = k$ and $(\mathbf{D^{-1}V})_\pi$ is full rank and solve $(\mathbf{D^{-1}V})_\pi\mathbf{x} = \mathbf{c}$ for $\mathbf{x}$ where the vector $\mathbf{c}$ consists of half integer elements. An arbitrary vertex ξ_i thus falls in the following set:

$$\xi_i \in \Big\{((\mathbf{D}^{-1}\mathbf{V})_\pi)^{-1}\mathbf{c} \;\Big|\; \pi \subseteq \{1,\ldots,n\}\,,\, |\pi| = k, \\ (\mathbf{D}^{-1}\mathbf{V})_\pi \text{ full rank}\,,\, \mathbf{c} - \frac{1}{2}\mathbf{1} \in \mathbb{Z}^k\Big\} \tag{19}$$

According to part b) of Theorem 2 not all such vertices need to be checked, since: $\|\mathbf{a}^*_\pi\| \leq \|\mathbf{a}^*\| \leq \psi$. Thus like in the case $k = 1$ we only need to check the vertices where

$$-(\psi + \frac{1}{2})\mathbf{1} < (\mathbf{D^{-1}V})_\pi\mathbf{x} < (\psi + \frac{1}{2})\mathbf{1}$$

and so

$$-(\lceil\psi\rceil + \frac{1}{2})\mathbf{1} \leq \mathbf{c} \leq (\lceil\psi\rceil + \frac{1}{2})\mathbf{1}$$

Now we can define the sets of all vertices of interest, Φ_π and their union Φ as

$$\Phi_\pi = \left\{((\mathbf{D}^{-1}\mathbf{V})_\pi)^{-1}\mathbf{c} \;\middle|\; |\mathbf{c}| \leq (\lceil\psi\rceil + \frac{1}{2})\mathbf{1}\,,\, \mathbf{c} - \frac{1}{2}\mathbf{1} \in \mathbb{Z}^k\right\}$$

$$\pi \subseteq \{1\ldots n\}\,,\, |\pi| = k\,, (\mathbf{D}^{-1}\mathbf{V})_\pi \text{ full rank}$$

$$\Phi_\pi = \emptyset\,,\;\; \pi \subseteq \{1\ldots n\}\,,\, |\pi| = k\,, (\mathbf{D}^{-1}\mathbf{V})_\pi \text{ rank deficient}$$

$$\Phi = \bigcup_{\substack{\pi\subseteq\{1\ldots n\}\\ |\pi|=k}} \Phi_\pi$$

In the next phase of the algorithm we use this set of vertices to find a set of interior points of polytopes of interest. It is not clear to us how to find exactly one point per polytope. The major difficulty is to identify which vertex belongs to which polytope. But for our main goal of showing a polynomial bound on complexity, this is immaterial.

In order to find *at least* one point in the interior of each polytope, we then consider all possible combinations of k + 1 vertices. Assuming they form a simplex in $\mathbb{R}^k$, we can then find an interior point of this simplex by taking the average of the k + 1 vertices. Note that if the chosen vertices lie in a k-dimensional space, then they do not form a simplex. Nonetheless the algorithm can check the average of these points, even if the theorem does not consider it a potential minimizer.

Since any convex polytope can be decomposed into simplexes, an interior point of all the polytopes must have been found in this process. The last step is to check the value of f over all these candidate points. In line with the theorem, one also has to separately check all the standard unit vectors.

This is summarized in Algorithm 2.

Algorithm 2 Finding the optimal coefficient vector, the general case

```
Input: Gram matrix $\mathbf{G}$ and its $DP^k$ decomposition, $\mathbf{D}$ and $\mathbf{V}$ matrices as defined
Output: $\mathbf{a}^*$

Initialization:
$\mathbf{u}_i :=$ standard unit vector in the direction of $i$-th axis
$\lambda_{\min} :=$ minimum eigenvalue of $\mathbf{G}$
$G_{\min} :=$ minimum diagonal element of $\mathbf{G}$
$\psi := \sqrt{\frac{G_{\min}}{\lambda_{min}}}$
$\Phi = \emptyset$
$f(\mathbf{a}) := \mathbf{a^T Ga}$
$f_{\min} = G_{\min}$
$\mathbf{a}^* = \mathbf{u}_{\arg\min(diag(\mathbf{G}))}$

Phase 1:
for all $\pi \subseteq \{1,\ldots,n\}$, $|\pi| = k$, and $(\mathbf{D^{-1}V})_\pi$ full rank do
    for all $\mathbf{c}$ , $|\mathbf{c}| \leq (\lceil\psi\rceil + \frac{1}{2})\mathbf{1}$ and $\mathbf{c} - \frac{1}{2}\mathbf{1} \in \mathbb{Z}^k$ do
        calculate $\mathbf{x} = ((\mathbf{D^{-1}V})_\pi)^{-1}\mathbf{c}_\pi$
        Set $\Phi = \Phi \cup \{\mathbf{x}\}$
    end for
end for

Phase 2:
for all possible choices of $k+1$ points in $\Phi$ do
    calculate $\mathbf{p}$ = average of the points
    calculate $\mathbf{b} = \mathbf{D^{-1}Vp}$
    calculate $\mathbf{a} = \lceil\mathbf{b}\rfloor$
    if $f(\mathbf{a}) < f_{\min}$ AND $\mathbf{a}$ is not the all zero vector then
        set $\mathbf{a}^* = \mathbf{a}$
        set $f_{\min} = f(\mathbf{a})$
end if
end for
return $\mathbf{a}^*$
```

A. Complexity Analysis

The running time of the algorithm is clearly dominated by phase 2, where all possible $k+1$ combinations of the points found in phase 1 are checked as potential vertices of a simplex. First we count the number of points found in phase 1. This number is given by

$$\sum_{\substack{\pi\subset\{1\ldots n\}\\ |\pi|=k}} (2\lceil\psi\rceil + 2)^k = \binom{n}{k}(2\lceil\psi\rceil + 2)^k \\ \leq \frac{n^k}{k!}(2\lceil\psi\rceil + 2)^k = \frac{(2n(\lceil\psi\rceil + 1))^k}{k!} \tag{**}$$

The number of loops in phase 2 is the number of possible choices of $k+1$ points out of all points found in the phase 1. It can be upper bounded using equation (**):

$$\binom{\frac{(2n(\lceil\psi\rceil+1))^k}{k!}}{k+1} \le \frac{(2n(\lceil\psi\rceil+1))^{k(k+1)}}{(k!)^{k+1}(k+1)!}$$

In order to find the complexity of the algorithm, we need to multiply this number of loops with the running time of each loop. Inside the loop, calculating the vector $\mathbf{b}$ can be done in $O(kn)$ and $f(\mathbf{a})$ can also be calculated in $O(kn)$ operations. Thus the complexity of the algorithm is

$$O\left(kn\frac{(2n(\lceil\psi\rceil+1))^{k(k+1)}}{(k!)^{k+1}(k+1)!}\right) = O\left(\frac{n\,(2n(\lceil\psi\rceil+1))^{k(k+1)}}{(k!)^{k+2}}\right)$$

Since this expression is a polynomial function of $\lceil\psi\rceil = \left\lceil\sqrt{\frac{G_{min}}{\lambda_{min}}}\right\rceil$, we conclude that as long as $\frac{G_{min}}{\lambda_{min}}$ is upper-bounded by a polynomial function of n the complexity of the algorithm is polynomial in n. As we saw, in the case of MIMO Compute-and-Forward, we have $\psi = \sqrt{1+P\gamma_{max}^2}$. Also, we know that $\gamma_{max}^2 \le trace(\mathbf{HH}^T) \le nkH_{max}^2$, where H_{max} is the maximum of all $|H_{ij}|$ values. Thus, if H_{max} is bounded by a polynomial function of n, the algorithm will be of polynomial complexity. This is truly the case in all practical models, such as Rayleigh fading or any model with bounded channel coefficients.

VI. PROOF OF THEOREM 2

part a)

First note that we can rewrite $f(\mathbf{a}) = \mathbf{a^TGa}$ as follows:

$$f(\mathbf{a}) = \sum_{i=1}^{n}(D_{ii}-P_{ii})a_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{i-1}P_{ij}a_ia_j$$

Assume that we already know the optimal value for all a_i^* elements except for one element, a_j. Note that f is a convex parabola in a_j (this is because $D_{jj}-P_{jj} = G_{jj}$ is a diagonal element of a positive definite matrix) thus the optimal integer value for a_j is the closest integer to its optimal real value. By taking partial derivative with respect to a_j, the optimal real value of a_j is easily seen to be equal to

$$\frac{\sum_{i=1,i\neq j}^{n}P_{ij}a_i^*}{D_{jj}-P_{jj}}.$$

Taking the closest integer to the real valued solution, we find:

$$\Rightarrow a_j^* = \left\lceil\frac{\sum_{i=1,i\neq j}^{n}P_{ij}a_i^*}{D_{jj}-P_{jj}}\right\rceil \quad OR \quad a_j^* = \left\lfloor\frac{\sum_{i=1,i\neq j}^{n}P_{ij}a_i^*}{D_{jj}-P_{jj}}\right\rfloor \qquad \text{(I)}$$

Due to the symmetry of the parabola, both functions return equally correct solutions for a_j^*.

Note that this expression must be true for any j: If for $\mathbf{a}^*$ and for some j, a_j^* does not satisfy at least one of these two equations, we can achieve a strictly smaller value over f by replacing a_j^* with the value given above, and so $\mathbf{a}^*$ cannot be optimal. The only situation where this logic fails is when in the optimal vector we have: $a_i^* = 0$, $i = 1\dots n$, $i \neq j$. In this case, replacing the value of a_j^* with its round expression will result in the all zero vector, $\mathbf{a}^* = \mathbf{0}$. Hence, the case where $\mathbf{a}^*$ is zero except in one element requires separate attention, as pointed out by the theorem. Under this assumption, $f(\mathbf{a}^*) = G_{jj}a_j^{*2}$. Thus it must be that $|a_j^*| = 1$, and so $\mathbf{a}^*$ is a standard unit vector, up to a sign.

Returning to the general case of $\mathbf{a}^*$ and from (I) we have that:

$$a_j^* + \frac{1}{2} \ge \frac{\sum_{i=1,i\neq j}^{n}P_{ij}a_i^*}{D_{jj}-P_{jj}} \quad , \; and \qquad \text{(II)}$$

$$a_j^* - \frac{1}{2} \le \frac{\sum_{i=1,i\neq j}^{n}P_{ij}a_i^*}{D_{jj}-P_{jj}} \qquad \text{(III)}$$

Starting with equation (II), we multiply both sides by the denominator, and add the term $a_j^*P_{jj}$ to obtain:

$$(a_j^* + \frac{1}{2})D_{jj} \ge \sum_{i=1}^{n}P_{ij}a_i^* + \frac{1}{2}P_{jj}$$

Dropping the non-negative term $\frac{1}{2}P_{jj}$ we conclude

$$(a_j^* + \frac{1}{2})D_{jj} \ge \sum_{i=1}^{n}P_{ij}a_i^*$$

Now we show that this inequality is strict, even if $P_{jj} = 0$. Due to the fact that $\mathbf{P}$ is positive semi-definite, we must have that if $P_{jj} = 0$ then $P_{ij} = 0$, $i = 1\dots n$. Thus in that case, the inequality turns into $(a_j^* + \frac{1}{2})D_{jj} \ge 0$. But we have that a_j^* is an integer and $D_{jj} - P_{jj} > 0$ thus $D_{jj} > 0$. So, $(a_j^* + \frac{1}{2})D_{jj}$ cannot be equal to zero and this inequality must be strict. As a result, we have:

$$(a_j^* + \frac{1}{2})D_{jj} > \sum_{i=1}^{n}P_{ij}a_i^*$$
$$\Rightarrow (a_j^* + \frac{1}{2}) > \frac{\sum_{i=1}^{n}P_{ij}a_i^*}{D_{jj}} \; , \; j = 1\dots n$$

Writing this inequality in vector format, we obtain

$$\mathbf{a}^* + \frac{1}{2}\mathbf{1} > \mathbf{D^{-1}P}^T\mathbf{a}^* = \mathbf{D^{-1}V}(\mathbf{V}^T\mathbf{a}^*) \qquad \text{(IV)}$$

In a similar fashion one can show that equation (III) results in

$$\Rightarrow \mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D^{-1}P}^T\mathbf{a}^* = \mathbf{D^{-1}V}(\mathbf{V}^T\mathbf{a}^*) \qquad \text{(V)}$$

Defining $\mathbf{x} = \mathbf{V}^T\mathbf{a}^*$, it follows from (IV) and (V) that

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D^{-1}Vx} < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$$

$$\Rightarrow \mathbf{a}^* = \lceil\mathbf{D^{-1}Vx}\rfloor$$

This completes the proof of part a).

part b)

First note that

$$f(\mathbf{a}^*) = \mathbf{a}^{*T}\mathbf{G}\mathbf{a}^* \geq \lambda_{min}\|\mathbf{a}^*\|^2$$

By simply choosing $\mathbf{a}$ to be the i-th standard unit vector, we have $f(\mathbf{a}) = G_{ii}$. Thus:

$$G_{min} \geq f(\mathbf{a}^*) \geq \lambda_{min}\|\mathbf{a}^*\|^2$$

from which we can conclude

$$\|\mathbf{a}^*\| \leq \sqrt{\frac{G_{min}}{\lambda_{min}}}$$

which is the claim made by part b) of the theorem.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we introduced an exact algorithm of polynomial complexity for solving the special case of SLV problem which appears in the context of Compute-and-Forward. We then generalized our results to the case of MIMO Compute-and-Forward. There are several possibilities to continue this work. The results may be extendable to more general lattices. Furthermore such Gram matrices may be used as a point of reference for approximating the shortest vector in a wider range of lattices. Finally, we conjecture that particular choices of the matrix $\mathbf{V}$ in decomposition of the Gram matrix may allow for a more efficient algorithm by establishing simple relations between the x_i values.

ACKNOWLEDGMENT

We would like to thank Chien-Yi Wang for his interesting ideas which helped us in different stages of this work. We would also like to thank him, Robby McKilliam and Cheng Wang for their help with reviewing the paper. This work has been supported in part by the European Union under ERC Starting Grant 259530-ComCom.

REFERENCES

[1] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *Information Theory, IEEE Transactions on*, vol. 57, no. 10, pp. 6463–6486, 2011.

[2] W. Nam, S.-Y. Chung, and Y. H. Lee, "Capacity bounds for two-way relay channels," in *Communications, 2008 IEEE International Zurich Seminar on*. IEEE, 2008, pp. 144–147.

[3] M. P. Wilson, K. Narayanan, H. D. Pfister, and A. Sprintson, "Joint physical layer coding and network coding for bidirectional relaying," *Information Theory, IEEE Transactions on*, vol. 56, no. 11, pp. 5641–5654, 2010.

[4] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[5] N. Gama and P. Q. Nguyen, "Finding short lattice vectors within mordell's inequality," in *Proceedings of the 40th annual ACM symposium on Theory of computing*. ACM, 2008, pp. 207–216.

[6] S. Khot, "Hardness of approximating the shortest vector problem in lattices," in *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS 2004)*. IEEE, 2004, pp. 126–135.

[7] D. Dadush, C. Peikert, and S. Vempala, "Enumerative lattice algorithms in any norm via m-ellipsoid coverings," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. IEEE, 2011, pp. 580–589.

[8] D. Dadush and D. Micciancio, "Algorithms for the densest sub-lattice problem," in *SODA*. SIAM, 2013, pp. 1103–1122.

[9] M. Alekhnovich, S. A. Khot, G. Kindler, and N. K. Vishnoi, "Hardness of approximating the closest vector problem with preprocessing," in *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*. IEEE, 2005, pp. 216–225.

[10] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*. Springer, 1999, vol. 290.

[11] ——, "Low-dimensional lattices. vi. voronoi reduction of three-dimensional lattices," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 436, no. 1896, pp. 55–68, 1992.

[12] R. G. McKilliam and A. Grant, "Finding short vectors in a lattice of voronoi's first kind," *arXiv preprint arXiv:1201.5154*, 2012.

[13] M. Hejazi and M. Nasiri-Kenari, "Simplified compute-and-forward and its performance analysis," *IET Communications*, vol. 7, no. 18, pp. 2054–2063, 2013.

[14] J. Richter, C. Scheunert, and E. Jorswieck, "An efficient branch-and-bound algorithm for compute-and-forward," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 77–82.

[15] B. Zhou and W. H. Mow, "A quadratic programming relaxation approach to compute-and-forward network coding design," in *Information Theory, 2014. ISIT 2014. IEEE International Symposium on*. IEEE, 2014, pp. 2296–2300.

[16] R. G. McKilliam, I. V. L. Clarkson, W. D. Smith, and B. G. Quinn, "A linear-time nearest point algorithm for the lattice A n*," in *Information Theory and its Applications, 2008. ISITA 2008. International Symposium on*. IEEE, 2008, pp. 1–5.

[17] R. G. McKilliam, W. D. Smith, and I. V. L. Clarkson, "Linear-time nearest point algorithms for coxeter lattices," *Information Theory, IEEE Transactions on*, vol. 56, no. 3, pp. 1015–1022, 2010.

[18] J. Zhan, B. Nazer, M. Gastpar, and U. Erez, "MIMO compute-and-forward," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE, 2009, pp. 2848–2852.

[19] J. Saunderson, V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky, "Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1395–1416, 2012.

[20] A. Shapiro, "Weighted minimum trace factor analysis," *Psychometrika*, vol. 47, no. 3, pp. 243–264, 1982.