This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

# Column Reordering for Box-Constrained Integer Least Squares Problems

Stephen Breen
School of Computer Science
McGill University
Montreal, Quebec, Canada
Email: sbreen1@cs.mcgill.ca

Xiao-Wen Chang
School of Computer Science
McGill University
Montreal, Quebec, Canada
Email: chang@cs.mcgill.ca

***Abstract*—The box-constrained integer least squares problem (BILS) arises in MIMO wireless communications applications. Typically a sphere decoding algorithm (a tree search algorithm) is used to solve the problem. In order to make the search algorithm more efficient, the columns of the channel matrix in the BILS problem have to be reordered. To our knowledge, there are currently two algorithms for column reordering that provide the best known results. Both use all available information, but they were derived respectively from geometric and algebraic points of view and look different. In this paper we modify one to make it more computationally efficient and easier to comprehend. Then we prove the modified one and the other actually give the same column reordering in theory. Finally we propose a new mathematically equivalent algorithm, which is more computationally efficient and is still easy to understand.**

## I. Introduction

Given a real vector $y \in \mathbb{R}^m$ and a real matrix $H \in \mathbb{R}^{m\times n}$, integer vectors $l, u \in \mathbb{Z}^n$ with $l < u$, the box-constrained integer least squares (BILS) problem is defined as:

$$\min_{x\in\mathcal{B}} \|y - Hx\|_2, \tag{1}$$

where $\mathcal{B} = \mathcal{B}_1 \times \cdots \times \mathcal{B}_n$ with $\mathcal{B}_i = \{x_i \in \mathbb{Z} : l_i \le x_i \le u_i\}$. This problem arises in wireless communications applications such as MIMO signal decoding. In this paper, we assume that $H$ has full column rank. The set $\{w = Hx : x \in \mathbb{Z}^n\}$ is referred to as the lattice generated by $H$.

Let $H$ have the QR factorization

$$H = [Q_1, Q_2]\begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $[\underset{n}{Q_1}, \underset{m-n}{Q_2}] \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular. Then, with $\bar{y} = Q_1^T y$ the BILS problem (1) is reduced to

$$\min_{x\in\mathcal{B}} \|\bar{y} - Rx\|_2. \tag{2}$$

To solve this reduced problem sphere decoding search algorithms (see, e.g., [1], [2] and [3]) enumerate the elements in $\mathcal{B}$ in some order to find the optimal solution.

If we reorder the columns of $H$, i.e., we apply a permutation matrix $P$ to $H$ from the right, then we will obtain a different R-factor, resulting in different search speed. A few algorithms have been proposed to find $P$ to minimize the complexity of the search algorithms. In [1], the well-known V-BLAST column reordering strategy originally given in [4] was proposed for this purpose. In [3], the SQRD column reordering strategy originally presented in [5] for the same purpose as V-BLAST, was proposed for this purpose. Both strategies use only the information of the matrix $H$.

In [6], Su and Wassell considered the geometry of the BILS problem for the case that $H$ is nonsingular and proposed a new column reordering algorithm (to be called the SW algorithm from here on for convenience) which uses all information of the BILS problem (1). Unfortunately, in our point of view, the geometric interpretation of this algorithm is hard to understand. Probably due to page limit, the description of the algorithm is very concise, making efficient implementation difficult for ordinary users.

In this paper we will give some new insight of the SW algorithm from an algebraic point of view. We will make some modifications so that the algorithm becomes more efficient and easier to understand and furthermore it can handle a general full column rank $H$.

Independently Chang and Han in [3] proposed another column reordering algorithm (which will be referred to as CH). Their algorithm also uses all information of (1) and the derivation is based on an algebraic point of view. It is easy to see from the equations in the search process exactly what the CH column reordering is doing and why we should expect a reduced complexity in the search process. The detailed description of the CH column reordering is given in [3] and it is easy for others to implement the algorithm. But our numerical tests indicated CH has a higher complexity than SW, when SW is implemented efficiently. Our numerical tests also showed that CH and SW *almost* always produced the same permutation matrix $P$.

In this paper, we will show that the CH algorithm and the (modified) SW algorithm give the same column reordering in theory. This is interesting because both algorithms were derived through different motivations and we now have both a geometric justification and an algebraic justification for why the column reordering strategy should reduce the complexity of the search. Furthermore, using the knowledge that certain steps in each algorithm are equivalent, we can combine the best parts from each into a new algorithm. The new algorithm

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

has a lower flop count than either of the originals. This is important to the successive interference cancellation decoder, which computes a suboptimal solution to (1). The new algorithm can be interpreted in the same way as CH, so it is easy to understand.

In this paper, $e_i$ denotes the $i^{th}$ column of the identity matrix $I$. For a set of integer numbers $\mathcal{S}$ and real number $x$, $\lfloor x \rceil_{\mathcal{S}}$ denotes the nearest integer in $\mathcal{S}$ to $x$ and if there is a tie it denotes the one which has smaller magnitude. For $z \in \mathcal{S}$, $\mathcal{S} \backslash z$ denotes $\mathcal{S}$ after $z$ is removed. We sometimes use MATLAB-like notation for matrices and vectors, e.g., $A_{1:m,1:n}$ denotes the matrix formed by the first $m$ rows and $n$ columns of the matrix $A$ and $A_{:,1:n}$ denote the matrix formed by the first $n$ columns of $A$. The $j^{th}$ column of a matrix $A$ is demoted either by $a_j$ or $A_{:,j}$.

## II. Search Process

Both CH and SW column reordering algorithms use ideas that arise from the search process. Before the column reorderings are introduced, it is important to have an understanding of the sphere decoding search process.

Consider the ILS problem (2). We would like to enumerate the elements in $\mathcal{B}$ in an efficient manner in order to find the solution $x$. One such enumeration strategy is described in [3]. We will now describe it briefly.

Suppose that the solution satisfies the following bound,

$$\|\bar{y} - Rx\|_2^2 < \beta. \tag{3}$$

There are a few ways to choose a valid initial value for $\beta$, see, e.g., [3]. The inequality (3) defines an ellipsoid in terms of $x$ or a hyper-sphere in terms of the lattice point $w = Rx$ with radius $\beta$. Define

$$c_k = (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj} x_j)/r_{kk},\ k = n, n-1, \ldots, 1, \tag{4}$$

where when $k = n$ the sum in the right hand side does not exist. Then (3) can be rewritten as

$$\sum_{k=1}^{n} r_{kk}^2 (x_k - c_k)^2 < \beta,$$

which implies the following set of inequalities:

$$\text{level } k: \quad r_{kk}^2 (x_k - c_k)^2 < \beta - \sum_{i=k+1}^{n} r_{ii}^2 (x_i - c_i)^2, \tag{5}$$

for $k = n, n-1, \ldots, 1$.

We begin the search process at level $n$. Choose $x_n = \lfloor c_n \rceil_{\mathcal{B}_n}$, the nearest integer in $\mathcal{B}_n$ to $c_n$. If the inequality (5) with $k = n$ is not satisfied, it will not be satisfied for any integer, this means $\beta$ was chosen to be too small, it must be enlarged. With $x_n$ fixed, we can move to level $n-1$ and choose $x_{n-1} = \lfloor c_{n-1} \rceil_{\mathcal{B}_{n-1}}$ with $c_{n-1}$ calculated as in (4). At this point it is possible that the inequality (5) is no longer satisfied. If this is the case, we must move back to level $n$ and choose $x_n$ to be the second nearest integer to $c_n$. We will continue this procedure until we reach level 1, moving back a level if ever the inequality for the current level is no longer satisfied. When we reach level 1, we will have found an integer point $\hat{x}$. We then update $\beta = \|\bar{y} - R\hat{x}\|_2^2$ and try to find a better integer point which satisfies the box-constraint in the new ellipsoid. Finally in the search process, when we can no longer find any $x_n$ to satisfy (5) with $k = n$, the search process is complete and the last integer point $\hat{x}$ found is the solution.

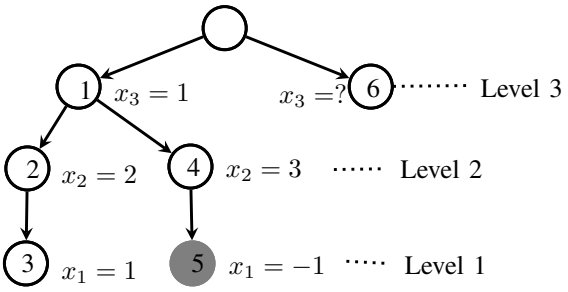



Fig. 1. An example of the search process with solution $x = [-1, 3, 1]^T$.

The above search process is actually a depth-first tree search, see Fig. 1, where the number in a node denote the step number at which the node is encountered.

## III. Column Reordering

In this section we introduce the two orginal column reordering algorithms, CH and SW and explain their motivations. We give some new insight on SW and propose a modified version. We also give a complexity analysis for both algorithms.

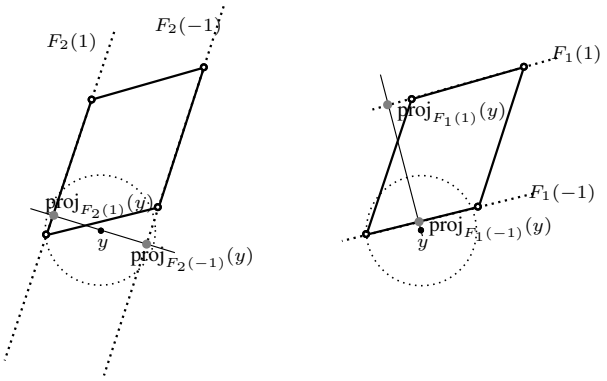
### A. Chang and Han's Algorithm

The CH algorithm first computes the QR factorization $H$, then tries to reorder the columns of $R$. The motivation for this algorithm comes from observing equation (5). If the inequality is false we know that the current choice for the value of $x_k$ given $x_{k+1:n}$ are fixed is incorrect and we prune the search tree. We would like to choose the column permutations so that it is likely that the inequality will be false at higher levels in the search tree. The CH column reordering strategy does this by trying to maximize the left hand side of (5) with large values of $|r_{kk}|$ and minimize the right hand side by making $|r_{kk}(x_k - c_k)|$ large for values of $k = n, n-1, \ldots, 1$.

Here we describe step 1 of the CH algorithm, which determines the last column of the final $R$ (or equivalently the last column of the final $H$). Subsequent steps are the same but are applied to a subproblem that is one dimension smaller. In step 1, for $i = 1, \ldots, n$ we interchange columns $i$ and $n$ of $R$ (thus entries of $i$ and $n$ in $x$ are also swapped), then return $R$ to upper-triangular by a series of Givens rotations applied to $R$ from the left, which are also applied to $\bar{y}$. To avoid confusion, we denote the new $R$ by $\hat{R}$ and the new $\bar{y}$ by $\hat{y}$. We then compute $c_n = \hat{y}_n / \hat{r}_{n,n}$ and

$$x_i^c = \arg \min_{x_i \in \mathcal{B}_i} |\hat{r}_{nn}(x_i - c_n)| = \lfloor c_n \rceil_{\mathcal{B}_i}, \tag{6}$$

where the superscript $c$ denotes the CH algorithm. Let $\bar{x}_i^c$ be the second closest integer in $\mathcal{B}_i$ to $c_n$, i.e., $\bar{x}_i^c = \lfloor c_n \rceil_{\mathcal{B}_i \backslash x_i^c}$.




(a) H - original column ordering (b) H - columns swapped

Fig. 2. Geometry of the search with two different column ordering.

Define

$$\text{dist}_i^c = |\hat{r}_{nn}(\bar{x}_i^c - c_n)|, \tag{7}$$

which represents the partial residual given when $x_i$ is taken to be $\bar{x}_i^c$. Let $j = \arg\max_i \text{dist}_i^c$. Then column $j$ of the original $R$ is chosen to be the $n^{th}$ column of the final $R$. With the corresponding updated upper triangular $R$ and $\bar{y}$ (here for convenience we have removed hats), the algorithm then updates $\bar{y}_{1:n-1}$ again by setting $\bar{y}_{1:n-1} := \bar{y}_{1:n-1} - r_{1:n-1,n}x_j$ where $x_j = x_j^c$. Choosing $x_j$ to be $x_j^c$ here is exactly the same as what the search process does. We then continue to work on the subproblem

$$\min_{\tilde{x}\in\mathbb{Z}^{n-1}} \|\bar{y}_{1:n-1} - R_{1:n-1,1:n-1}\tilde{x}\|_2 , \tag{8}$$

where $\tilde{x} = [x_1, \ldots, x_{j-1}, x_n, x_{j+1}, \ldots x_{n-1}]^T$ satisfies the corresponding box constraint. The pseudocode of the CH algorithm is given in Algorithm 1.

To determine the last column, CH finds the permutation to maximize $|r_{nn}(\bar{x}_i^c - c_n)|$. Using $\bar{x}_i^c$ instead of $x_i^c$ ensures that $|\bar{x}_i^c - c_n|$ is never less than 0.5 but also not very large. This means that usually if $|r_{nn}(\bar{x}_i^c - c_n)|$ is large, $|r_{nn}|$ is large as well and the requirement to have large $|r_{nn}|$ is met. Using $x_i^c$ would not be a good choice because $|x_i^c - c_n|$ might be very small or even 0, then column $i$ would not be chosen to be column $n$ even if the corresponding $|r_{nn}|$ is large and on the contrary a column with small $|r_{nn}|$ but large $|x_i^c - c_n|$ may be chosen.

Now we will consider the complexity of CH. The significant cost comes from line 9 in Algorithm 1, which requires $6(k-i)^2$ flops. If we sum this cost over all loop iterations and add the cost of the QR factorization by Householder transformations, we get a total complexity of $0.5n^4 + 2mn^2$ flops.

### *B. Su and Wassell's Algorithm*

The motivation for the SW algorithm comes from examining the geometry of the search process.

Fig. 2 shows a 2-D BILS problem; 2(a) represents the original column ordering and 2(b) is after the columns have been swapped.

**Algorithm 1** CH Algorithm - Returns $p$, the column permutation vector

1: $p := 1 : n$
2: $p' := 1 : n$
3: Compute the QR factorization of $H$: $\begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} H = \begin{bmatrix} R \\ 0 \end{bmatrix}$ and compute $\bar{y} := Q_1^T y$
4: **for** $k = n$ to 2 **do**
5: $\quad maxDist := -1$
6: $\quad$ **for** $i = 1$ to $k$ **do**
7: $\quad\quad \hat{y} := \bar{y}_{1:k}$
8: $\quad\quad \hat{R} := R_{1:k,1:k}$
9: $\quad\quad$ swap columns $i$ and $k$ of $\hat{R}$, return it to upper triangular with Givens rotations, also apply the Givens rotations to $\hat{y}$
10: $\quad\quad x_i^c := \lfloor \hat{y}_k / \hat{r}_{k,k} \rceil_{\mathcal{B}_i}$
11: $\quad\quad \bar{x}_i^c := \lfloor \hat{y}_k / \hat{r}_{k,k} \rceil_{\mathcal{B}_i \setminus x_i^c}$
12: $\quad\quad dist_i^c := |\hat{r}_{k,k}\bar{x}_i^c - \hat{y}_k|$
13: $\quad\quad$ **if** $dist_i^c > maxDist$ **then**
14: $\quad\quad\quad maxDist := dist_i^c$
15: $\quad\quad\quad j := i$
16: $\quad\quad\quad R' := \hat{R}$
17: $\quad\quad\quad y' := \hat{y}$
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: $\quad p_k := p'_j$
21: $\quad$ Interchange the intervals $\mathcal{B}_k$ and $\mathcal{B}_j$
22: $\quad$ Intechange entries $k$ and $j$ in $p'$
23: $\quad R_{1:k,1:k} := R'$
24: $\quad \bar{y}_{1:k} := y' - R'_{1:k,k}x_j^c$
25: **end for**
26: $p_1 := p'_1$

In the SW algorithm $H = [h_1, \ldots, h_n]$ is assumed to be square and non-singular. Let

$$G = [g_1, \ldots, g_n] = H^{-T}.$$

For any integer $\alpha$, [6] defines the affine sets, $F_i(\alpha) = \{w \mid g_i^T(w - h_i\alpha) = 0\}$. The lattice points generated by $H$ occur at the intersections of these affine sets. Let the orthogonal projection of a vector $s$ onto a vector $t$ be denoted as $\text{proj}_t(s)$, then the orthogonal projection of some vector $s$ onto $F_i(\alpha)$ is $\text{proj}_{F_i(\alpha)}(s) = s - \text{proj}_{g_i}(s - h_i\alpha)$. Therefore the orthogonal distance between $s$ and $F_i(\alpha)$ is $\text{dist}(s, F_i(\alpha)) = \|s - \text{proj}_{F_i(\alpha)}(s)\|_2$. In [6], the points labeled $\text{proj}_{F_2(1)}(y)$ and $\text{proj}_{F_2(-1)}(y)$ in Fig. 2 are called residual targets and "represent the components [of $y$] that remain after an orthogonal part has been projected away."

Note that $F_2(\alpha)$ in Fig. 2 is a sublattice of dimension 1. Algebraically it is the lattice generated by $H$ with column 2 removed. It can also be thought of as a subtree of the search tree where $x_2 = \alpha$ has been fixed. In the first step of the search process for a general case, $x_n$ is chosen to be $x_n = \arg\min_{\alpha\in\mathcal{B}_n} \text{dist}(y, F_n(\alpha))$; thus $F_n(x_n)$ is the nearest affine set to $y$. Actually the value of $x_n$ is identical to $\lfloor c_n \rceil_{\mathcal{B}_n}$ given

in Section II, which will be proved later. Then $y$ is updated as $y := \text{proj}_{F_n(x_n)}(y) - h_n x_n$. If we look at Fig. 2, we see that the projection $\text{proj}_{F_n(x_n)}(y)$ moves $y$ onto $F_n(x_n)$, while the subtraction of $h_n x_n$ algebraically fixes the value of $x_n$. This is necessary because in subsequent steps we will not consider the column $h_n$.

We now apply the same process to the new $n-1$ dimensional search space $F_n(x_n)$. If at some level $i$, $\min_{\alpha\in\mathcal{B}_i} \text{dist}(y, F_i(\alpha))$ exceeds the current search radius, we must move back to level $i+1$. When the search process reaches level 1 and fixes $x_1$, it updates the radius to $\text{dist}(y, F_1(x_1))$ and moves back up to level 2.

Note that this search process is mathematically equivalent to the one described in section II; the difference is that it does projections because the generator matrix is not assumed to be upper-triangular. Computationally the former is more expensive than the latter.

To see the motivation of the SW algorithm for choosing a particular column ordering, consider Fig. 2. Suppose the search algorithm has knowledge of the residual for the optimal solution (the radius of the circle in the diagram). With the column ordering chosen in (a), there are two possible choices for $x_2$, leading to the two dashed lines $F_2(-1)$ and $F_2(1)$ which cross the circle. This means that we will need to find $x_1$ for both of these choices before we can determine which one leads to the optimum solution. In (b), there is only one possible choice for $x_1$, leading to the only dashed line $F_1(-1)$ which crosses the circle, meaning we only need to find $x_2$ to find the optimum solution. Since the projection resulting from the correct choice of $x_2$ will always be within the sphere, it makes sense to choose the ordering which maximizes the distance to the second best choice for $x_2$ in hopes that the second nearest choice will result in a value for $\min_{\alpha\in\mathcal{B}_2} \text{dist}(y, F_2(\alpha))$ outside the sphere and the dimensionality can be reduced by one. For more detail on the geometry, see [6].

The following will give an overview of the SW algorithm as given in [6] but described in a framework similar to what was used to describe CH. In the first step to determine the last column, for each $i = 1, \ldots, n$, we compute

$$x_i^s = \arg\min_{\alpha\in\mathcal{B}_i} \text{dist}(y, F_i(\alpha)) = \arg\min_{\alpha\in\mathcal{B}_i} |y^T g_i - \alpha| = \left\lfloor y^T g_i \right\rceil_{\mathcal{B}_i}, \tag{9}$$

where the superscript $s$ stands for the SW algorithm. Let $\bar{x}_i^s$ be the second closest integer in $\mathcal{B}_i$ to $y^T g_i$, i.e., $\bar{x}_i^s = \left\lfloor y^T g_i \right\rceil_{\mathcal{B}_i \setminus x_i^s}$. Let $j = \arg\max_i \text{dist}(y, F_i(\bar{x}_i^s))$. Then SW chooses column $j$ as the last column of the final reordered $H$, updates $y$ by setting $y := \text{proj}_{F_j(x_j^s)}(y) - h_j x_j^s$ and updates $G$ by setting $g_i := \text{proj}_{F_j(0)}(g_i)$ for all $i \neq j$. After $G$ and $y$ have been updated, the algorithm continues to find column $n-1$ in the same way etc. The pseudo-code of the SW algorithm is given in Algorithm 2.

[6] did not say how to implement the algorithm and did not give a complexity analysis. The parts of the cost we must consider for implementation occur in lines 9 and 19. Note that $\text{dist}(y, F_i(\bar{x}_i^s)) = \|\text{proj}_{g_i}(y - h_i \bar{x}_i^s)\|_2$ and $\text{proj}_{F_j(0)}(g_i) = g_i - \text{proj}_{g_i} g_i$, where $\text{proj}_{g_i} = g_i g_i^\dagger = g_i g_i^T / \|g_i\|^2$. A naive implementation would first compute $\text{proj}_{g_i}$, requiring $n^2$ flops, then compute $\|\text{proj}_{g_i}(y - h_i \bar{x}_i^s)\|_2$ and $g_i - \text{proj}_{g_i} g_i$, each requiring $2n^2$ flops. Summing these costs over all loop iterations we get a total complexity of $2.5n^4$ flops. In the next subsection we will simplify some steps in Algorithm 2 and show how to implement them efficiently.

**Algorithm 2** SW Algorithm - Returns $p$, the column permutation vector

```
1:  p := 1 : n
2:  p' := {1, 2, ..., n}
3:  G := H^{-T}
4:  for k = n to 2 do
5:      maxDist := -1
6:      for i ∈ p' do
7:          x_i^s := ⌊y^T g_i⌉_{B_i}
8:          x̄_i^s := ⌊y^T g_i⌉_{B_i \ x_i^s}
9:          dist_i^s := dist(y, F_i(x̄_i^s))
10:         if dist_i^s > maxDist then
11:             maxDist := dist_i^s
12:             j := i
13:         end if
14:     end for
15:     p_k := j
16:     p' := p' \ j
17:     y := proj_{F_j(x_j^s)}(y) - h_j x_j^s
18:     for i ∈ p' do
19:         g_i := proj_{F_j(0)}(g_i)
20:     end for
21: end for
22: p_1 := p'
```

### C. Algebraic Interpretation and Modifications of SW

In this section we give new algebraic interpretation of some steps in Algorithm 2, simplify some key steps to improve the efficiency, and extend the algorithm to handle a more general case. All line numbers refer to Algorithm 2.

First we show how to efficiently compute $\text{dist}_i^s$ in line 9. Observing that $g_i^T h_i = 1$, we have

$$\text{dist}_i^s = \|g_i g_i^\dagger (y - h_i \bar{x}_i^s)\|_2 = |y^T g_i - \bar{x}_i^s| / \|g_i\|_2. \tag{10}$$

Note that $y^T g_i$ and $\bar{x}_i^s$ have been computed in lines 7 and 8, respectively. So the main cost of computing $\text{dist}_i^s$ is the cost of computing $\|g_i\|_2$, requiring only $2n$ flops. For $k = n$ in Algorithm 2, $y^T g_i = y^T H^{-T} e_i = (H^{-1} y)^T e_i$, i.e., $y^T g_i$ is the $i^{th}$ entry of the real solution for $Hx = y$. The interpretation can be generalized to a general $k$.

In line 19 Algorithm 2,

$$\begin{aligned} g_i^{\text{new}} &\equiv \text{proj}_{F_j(0)}(g_i) \\ &= (I - \text{proj}_{g_j}) g_i = g_i - g_j (g_j^T g_i / \|g_j\|_2^2). \end{aligned} \tag{11}$$

Using the last expression for computation needs only $4n$ flops (note that $\|g_j\|_2$ has been computed before, see (10)). We can actually show that the above is performing updating of $G$, the

Moore-Penrose generalized inverse of $H$ after we remove its $j^{th}$ column. For proof of this, see [7].

In line 17 of Algorithm 2,

$$y^{\text{new}} \equiv \text{proj}_{F_j(x_j^s)}(y) - h_j x_j^s = (y - g_j g_j^\dagger (y - h_j x_j^s)) - h_j x_j^s$$
$$= (I - \text{proj}_{g_j})(y - h_j x_j^s). \quad (12)$$

This means that after $x_j$ is fixed to be $x_j^s$, $h_j x_j^s$ is combined with $y$ (the same as CH does) and then the vector is projected to the orthogonal complement of the space spanned by $g_j$. We can show that this guarantees that the updated $y$ is in the subspace spanned by the columns of $H$ which have not been chosen. This is consistent with the assumption that $H$ is nonsingular, which implies that the original $y$ is in the space spanned by the columns of $H$. However, it is not necessary to apply the orthogonal projector $I - \text{proj}_{g_j}$ to $y - h_j x_j^s$ in (12). The reason is as follows. In Algorithm 2, $y^{\text{new}}$ and $g_i^{\text{new}}$ will be used only for computing $(y^{\text{new}})^T g_i^{\text{new}}$ (see line 7). But from (11) and (12)

$$(y^{\text{new}})^T g_i^{\text{new}} = (y - h_j x_j^s)^T (I - \text{proj}_{g_j})(I - \text{proj}_{g_j}) g_i$$
$$= (y - h_j x_j^s)^T g_i^{\text{new}}.$$

Therefore, line 17 can be replaced by $y := y - h_j x_j^s$. This not only simplifies the computation but also is much easier to interpret—after $x_j$ is fixed to be $x_j^s$, $h_j x_j^s$ is combined into $y$ as what the CH algorithm does. Let $H_{:,1:n-1}$ denote $H$ after its $j^{th}$ column is removed. We then continue to work on the subproblem

$$\min_{\check{x} \in \mathbb{Z}^{n-1}} \|y - H_{:,1:n-1}\check{x}\|_2, \quad (13)$$

where $\check{x} = [x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n]^T$ satisfies the corresponding box constraint. Here $H_{:,1:n-1}$ is not square. But there is no problem to handle it, see the next paragraph.

In [6], $H$ is assumed to be square and non-singular. In our opinion, this condition may cause confusion, since for each $k$ except $k = n$ in Algorithm 2, the remaining columns of $H$ which have not been chosen do not form a square matrix. Also the condition restricts the application of the algorithm to a general full column rank matrix $H$, unless we transform $H$ to a nonsingular matrix $R$ by the QR factorization. To extend the algorithm to a general full column rank matrix $H$, we need only replace line 3 by $G := (H^\dagger)^T$. This extension has another benefit. We mentioned before that the updating of $G$ in line 19 is actually the updating of the Moore-Pernrose generalized inverse of the matrix formed by the columns of $H$ which have not been chosen. So the extension makes all steps consistent.

To reliably compute $G$ for a general full column rank $H$, we can compute the QR factorization $H = Q_1 R$ by the Householder transformations and then solve the triangular system $RG^T = Q_1^T$ to obtain $G$. This requires $(5m - 4n/3)n^2$ flops. Another less reliable but more efficient way to do this is to compute $G = H(H^T H)^{-1}$. To do this efficiently we would compute the Cholesky factorization $H^T H = R^T R$ and solve $R^T R G^T = H^T$ for $G$ by using the triangular structure of $R$. The total cost for computing $G$ by this method can be shown to be $3mn^2 + \frac{n^3}{3}$. If $H$ is square and nonsingular, we would use the LU factorization with partial pivoting to compute $H^{-1}$ and the cost is $2n^3$ flops.

For the rest part of the algorithm if we use the simplification and efficient implementations mentioned above, we can show that it needs $4mn^2$ flops.

We see the modified SW algorithm is much more efficient than both the CH algorithm and the SW algorithm implemented in a naive way we mentioned in the previous subsection.

## IV. Equivalence of CH and SW

In this section we prove that CH and the modified SW produce the same set of permutations for a general full column rank $H$. To prove this it will suffice to prove that $x_i^s = x_i^c$, $\bar{x}_i^s = \bar{x}_i^c$, $\text{dist}_i^s = \text{dist}_i^c$ for $i = 1, \ldots, n$ in the first step which determines the last column of the final reordered $H$ and that the subproblems produced for the second step of each algorithm are equivalent.

Proving $x_i^s = x_i^c$ is not difficult. The only effect the interchange of columns $i$ and $n$ of $R$ in CH has on the real LS solution is that elements $i$ and $n$ of the solution are swapped. Therefore $x_i^c$ is just the $i^{th}$ element of the real LS solution rounded to the nearest integer in $\mathcal{B}_i$. Thus, with (6) and (9),

$$x_i^c = \lfloor (H^\dagger y)_i \rceil_{\mathcal{B}_i} = \lfloor e_i^T H^\dagger y \rceil_{\mathcal{B}_i} = \lfloor g_i^T y \rceil_{\mathcal{B}_i} = x_i^s. \quad (14)$$

Therefore we also have $\bar{x}_i^c = \bar{x}_i^s$.

In CH, after applying a permutation $P$ to swap columns $i$ and $n$ of $R$, we apply $V^T$, a product of the Givens rotations, to bring $R$ back to a new upper triangular matrix, denoted by $\hat{R}$, and also apply $V$ to $\bar{y}$, leading to $\hat{y} = V^T \bar{y}$. Thus $\hat{R} = V^T R P$ and $\hat{y} = V^T \bar{y} = V^T Q_1^T y$. Then $H = Q_1 R = Q_1 V \hat{R} P^T$, $H^\dagger = P \hat{R}^{-1} V^T Q_1^T$, $g_i = (H^\dagger)^T e_i = Q_1 V \hat{R}^{-T} P^T e_i = Q_1 V \hat{R}^{-T} e_n$, and $\|g_i\|_2 = \|\hat{R}^{-T} e_n\|_2 = 1/|\hat{r}_{nn}|$. Therefore, with (10) and (7)

$$\text{dist}_i^s = \frac{|y^T g_i - \bar{x}_i^s|}{\|g_i\|_2} = |\hat{r}_{nn}||y^T Q_1 V \hat{R}^{-T} e_n - \bar{x}_i^s| \quad (15)$$
$$= |\hat{r}_{nn}||\hat{y}_n/\hat{r}_{nn} - \bar{x}_i^s| = |\hat{r}_{nn}(c_n - \bar{x}_i^c)| = \text{dist}_i^c.$$

Now we consider the subproblem (8) in CH and the subproblem (13) in SW. We can easily show that $R_{1:n-1,1:n-1}$ in (8) is the $R$-factor of the QR factorization of $H_{:,1:n-1}P$, where $H_{:,1:n-1}$ is the matrix given in (13) and $P$ is a permutation matrix such that $\check{x} = P\tilde{x}$, and that $\bar{y}_{1:n-1}$ in (8) is the multiplication of the transpose of the $Q_1$-factor of the QR factorization of $H_{:,1:n-1}P$ and $y$ in (13). Thus the two subproblems are equivalent.

## V. New Algorithm

Now that we know the two algorithms are equivalent, we can take the best parts from both and combine them to form a new algorithm. The main cost in CH is to interchange the columns of $R$ and return it to upper-triangular form using Givens rotations. When we determine the $k^{th}$ column, we must do this $k$ times. We can avoid all but one of these column interchanges by computing $x_i^c$, $\bar{x}_i^c$ and $\text{dist}_i^c$ directly.

After the QR factorization of $H$, we solve the reduced ILS problem (2). We need only consider how to determine the last column of the final $R$. Other columns can be determined similarly. Here we use the ideas from SW. Let $G = R^{-T}$, which is lower triangular. By (14), we compute for $i = 1, \ldots, n$

$$x_i = \left\lfloor \bar{y}^T G_{:,i} \right\rceil_{\mathcal{B}_i} = \left\lfloor \bar{y}_{i:n}^T G_{i:n,i} \right\rceil_{\mathcal{B}_i}, \ \bar{x}_i = \left\lfloor \bar{y}_{i:n}^T G_{i:n,i} \right\rceil_{\mathcal{B}_i \setminus x_i},$$
$$\mathrm{dist}_i = |\bar{y}_{i:n}^T G_{i:n,i} - \bar{x}_i| / \|G_{i:n,i}\|_2.$$

Let $j = \arg\max_i \mathrm{dist}_i$. We take a slightly different approach to permuting the columns than was used in CH. Once $j$ is determined, we set $\bar{y}_{1:n-1} := \bar{y}_{1:n-1} - r_{1:n-1,j} x_j$. Then we simply remove the $j^{th}$ column from $R$, and restore it to upper triangular using Givens rotations. We then apply the same Givens rotations to the new $\bar{y}$. In addition, we must also update the inverse matrix $G$. This is very easy, we can just remove the $j^{th}$ column of $G$ and apply the same Givens rotations that were used to restore the upper triangular structure of $R$. To see this is true notice that removing column $j$ of $R$ is mathematically equivalent to rotating $j$ to the last column and shifting columns $j, j+1, \ldots, n$ to the left one position, since we will only consider columns $1, 2, \ldots, n-1$ in subsequent steps. Suppose $P$ is the permutation matrix which will permute the columns as described, and $V^T$ is the product of Givens rotations to restore $R$ to upper-triangular. Let $\hat{R} = V^T R P$ and set $\hat{G} = \hat{R}^{-T}$. Then

$$\hat{G} = (V^T R P)^{-T} = V^T R^{-T} P = V^T G P.$$

This indicates that the same $V$ and $P$, which are used to transform $R$ to $\hat{R}$, also transform $G$ to $\hat{G}$. Since $\hat{G}$ is lower triangular, it is easy to verify that $\hat{G}_{1:n-1,1:n-1} = \hat{R}^{-T}_{1:n-1,1:n-1}$. Both $\hat{R}_{1:n-1,1:n-1}$ and $\hat{G}_{1:n-1,1:n-1}$ will be used in the next step.

After this, as in the CH algorithm, we continue to work on the subproblem of size $n-1$. The advantages of using the ideas from CH are that we always have a lower triangular $G$ whose dimension is reduced by one at each step and the updating of $G$ is numerically stable as we use orthogonal transformations. We give the pseudocode of the new algorithm in Algorithm 3.

Here we consider the complexity analysis of our new algorithm. If we sum the costs in algorithm 3 over all loop iterations, we get a total of $\frac{7n^3}{3} + 2mn^2$ flops in the worst case. The worst case is very unlikely to occur, it arises when $j = 1$ each iteration of the outer loop. In the average case however, $j$ is around $k/2$ and we get an average case complexity of $\frac{4n^3}{3} + 2mn^2$ flops. In both cases, the complexity is less than the complexity of the modified SW algorithm.

**Algorithm 3** New algorithm

1: Compute the QR factorization of $H$ by Householder transformations: $\begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} H = \begin{bmatrix} R \\ 0 \end{bmatrix}$
and compute $\bar{y} := Q_1^T y$ $(2(m - n/3)n^2$ flops)
2: $G := R^{-T}$ $(\frac{n^3}{3}$ flops)
3: $p := 1 : n$
4: $p' := 1 : n$
5: **for** $k = n$ to 2 **do**
6: $maxDist := -1$
7: **for** $i = 1$ to $k$ **do**
8: $\alpha = y_{i:k}^T G_{i:k,i}$
9: $x_i := \lfloor \alpha \rceil_{\mathcal{B}_i}$ $(2(k-i)$ flops)
10: $\bar{x}_i := \lfloor \alpha \rceil_{\mathcal{B}_i \setminus x_i}$
11: $\mathrm{dist}_i = |\alpha - \bar{x}_i| / \|G_{i:k,i}\|_2$ $(2(k-i)$ flops)
12: **if** $dist_i > maxDist$ **then**
13: $maxDist := dist_i$
14: $j := i$
15: **end if**
16: **end for**
17: $p_k := p'_j$
18: Interchange the intervals $\mathcal{B}_k$ and $\mathcal{B}_j$
19: Interchange entries $k$ and $j$ in $p'$
20: Set $\bar{y} := \bar{y}_{1:k-1} - R_{1:k-1,j} x_j$
21: Remove column $j$ of $R$ and $G$, and return $R$ and $G$ to upper and lower triangular by Givens rotations, respectively, and then remove the last row of $R$ and $G$. The same Givens rotations are applied to $\bar{y}$.
$(6k(k-j)$ flops)
22: **end for**
23: $p_1 = p'_1$

## VI. Summary

We showed that two algorithms for the column reordering of the box-constrained ILS problem are equivalent. To do that, we modified one algorithm and gave new insight. We proposed a new algorithm by combining the best ideas from both of the originals. Our new algorithm is more efficient than either of the originals and is easy to implement and understand. Since the three algorithms are theoretically equivalent and were derived through different motivations, we now have both geometrical and algebraic motivations for the algorithms.

## References


[1] M. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[2] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier, "Soft-input soft-output lattice sphere decoder for linear channels," in *Proceedings of IEEE 2003 Global Communications Conference*, San Francisco, U.S.A., Dec. 2003, pp. 213–217.

[3] X.-W. Chang and Q. Han, "Solving box-constrained integer least-squares problems," *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 277–287, 2008.

[4] G. J. Foscini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[5] D. Wubben, R. Bohnke, J. Rinas, V. Kuhn, and K. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEEE Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

[6] K. Su and I. J. Wassell, "A new ordering for efficient sphere decoding," in *IEEE International Conference on Communications*, vol. 3, 2005, pp. 1906–1910.

[7] R. E. Cline, "Representations for the generalized inverse of a partitioned matrix," *Journal of the Society for Industrial and Applied Mathematics*, vol. 12, no. 3, pp. 588–600, Sep. 1964.