A NEW FAST GENERALIZED SPHERE DECODING ALGORITHM FOR UNDER-DETERMINED MIMO SYSTEMS

Xiao-Wen Chang, Xiaohua Yang

McGill University
School of Computer Science
3480 University Street, Montreal, Quebec H3A 2A7, Canada

ABSTRACT

Generalized sphere decoding (GSD) algorithms have been applied to decode the under-determined MIMO systems. It detects the transmitting vector by decoding a sequence of determined subproblems. In this paper a fast recursive GSD algorithm is proposed. This new algorithm can generate the sequence of determined subproblems in a more efficient way than the current algorithms. A column-reordering strategy for the channel matrix is incorporated into the reduction process of the new algorithm, which can significantly reduce the computational cost. Furthermore, a method to determine a good initial radius of the hyper-sphere is given. Numerical simulations show that the new recursive GSD algorithm can be significantly faster than the current algorithms.

1. INTRODUCTION

In Gaussian multi-input multi-output (MIMO) linear channels systems, the sphere decoding algorithms are often employed to detect the transmitting signal vector. Under some circumstances, the MIMO system may be under-determined. One such an application is the multiple-antenna communication systems where there are more transmitting antennas than receiving antennas. A typical approach to decoding under-determined MIMO system is to solve the following *under-determined integer least squares problem* (UILS)

$$\min_{\boldsymbol{x}\in\mathcal{I}_k^n} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{1}$$

where $\mathcal{I}_k$ is a 2^k-PAM signal set for some positive integer k: $\mathcal{I}_k = \{\text{odd integer } j : -2^k+1 \leq j \leq 2^k-1\}$, $\boldsymbol{y}\in\mathbb{R}^m$ is the received signal vector, $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ is a channel matrix with rank$(\boldsymbol{A})=m$, $\boldsymbol{x}\in\mathcal{I}_k^n$ is the transmitting vector.

A *generalized sphere decoding* (GSD) algorithm was first proposed by Damen et al [1] to solve (1). Recently Dayal and Varanasi [2] proposed another GSD algorithm, to be called Algorithm DV, which can significantly reduce the complexity by partitioning the candidate set into disjoint ordered subsets. Most recently Yang, Liu and He [3] proposed a new GSD algorithm, to be called Algorithm YLH, which is usually faster than Algorithm DV. All these algorithms mainly consider how to generate a sequence of determined sub-ILS problems. In this paper a recursive GSD algorithm will be proposed. The new algorithm can generate the sequence of determined sub-ILS problems in a more efficient way. The reduction process of the new algorithm incorporates a column-reordering strategy to decrease the generation time of the subproblems and the number of these subproblems to be solved. A method to determine a good initial radius of the hyper-sphere is also given, which overcomes a difficulty with current algorithms.

The rest of this contribution is organized as follows. In Section 2 we present our new GSD algorithm. Section 3 gives simulation results to show the efficiency of our recursive GSD algorithm. Finally a summary is given in Section 4.

2. A RECURSIVE GSD ALGORITHM

For simplicity, we assume that $k=1$ in $\mathcal{I}_k$, i.e., x_j take its value from $\{\pm1\}$ for $1\leq j\leq n$. The extension to the general case can be done similarly as in [2].

Suppose that $\boldsymbol{A}$ has the QR decomposition: $\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q}\in\mathbb{R}^{m\times m}$ is orthogonal and $\boldsymbol{R}\in\mathbb{R}^{m\times n}$ is an upper trapezoidal matrix (see, e.g., [4, Sec 5.2]). Let $\bar{\boldsymbol{y}}\triangleq\boldsymbol{Q}^T\boldsymbol{y}\in\mathbb{R}^m$ and $N\triangleq n-m+1$. Partition $\bar{\boldsymbol{y}}$, $\boldsymbol{R}$ and $\boldsymbol{x}$ as follows

$$\bar{\boldsymbol{y}}=\begin{array}{c}\left[\begin{array}{c}\bar{\boldsymbol{y}}^{(1)}\\ \bar{y}_m\end{array}\right]\begin{array}{l}m-1\\1\end{array}\end{array},\quad \boldsymbol{R}=\begin{array}{cc}\left[\begin{array}{cc}\boldsymbol{R}_1 & \boldsymbol{R}_2\\ \boldsymbol{0} & \boldsymbol{r}^T\end{array}\right] & \begin{array}{l}m-1\\1\end{array}\\ \begin{array}{cc}m-1 & N\end{array} & \end{array},\quad \boldsymbol{x}=\left[\begin{array}{c}\boldsymbol{x}^{(1)}\\ \boldsymbol{x}^{(2)}\end{array}\right]\begin{array}{l}m-1\\N\end{array}. \tag{2}$$

Then the problem (1) with $k=1$ is transformed to

$$\min_{\boldsymbol{x}\in\mathcal{I}_1^n}\|\bar{\boldsymbol{y}}-\boldsymbol{R}\boldsymbol{x}\|_2^2=\min_{\boldsymbol{x}^{(2)}\in\mathcal{I}_1^N}\Big[(\bar{y}_m-\boldsymbol{r}^T\boldsymbol{x}^{(2)})^2 + \min_{\boldsymbol{x}^{(1)}\in\mathcal{I}_1^{m-1}}\|(\bar{\boldsymbol{y}}^{(1)}-\boldsymbol{R}_2\boldsymbol{x}^{(2)})-\boldsymbol{R}_1\boldsymbol{x}^{(1)}\|_2^2\Big]. \tag{3}$$

Suppose that one has

$$\|\bar{\boldsymbol{y}}-\boldsymbol{R}\boldsymbol{x}\|_2^2\leq\beta^2 \tag{4}$$

This work was supported by NSERC of Canada grant RGPIN217191-03.

for some scalar β (we will discuss how to choose the initial β in Sect 2.3). Then it follows that

$$(\bar{y}_m - \boldsymbol{r}^T\boldsymbol{x}^{(2)})^2 \leq \beta^2. \tag{5}$$

From (3) we observe that the original problem can be solved in the following way. We first fix $\boldsymbol{x}^{(2)}$, then employ a conventional sphere decoding algorithm (see, e.g., [5] and [6]) to solve the sub-ILS problem within the brackets in (3). The problem can be solved by exhaustively trying every possible $\boldsymbol{x}^{(2)}$. This is the key idea of the GSD algorithm proposed in [1] (which uses a partition slightly different from (2)). But it is not efficient. Algorithm DV in [2] first determines a possible $\boldsymbol{x}^{(2)}$ which satisfies (5) in a more efficient way by a set partition approach, which will also be used here to find $\boldsymbol{x}^{(2)}$.

2.1. Solving the transformed problem (3)

We first give a new bound (see (6)) which is at least as tight as (5) by using the idea of [6]. Notice that $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|_2^2 = \sum_{i=1}^m(\bar{y}_i - \sum_{j=i}^n r_{ij}x_j)^2$ and each $x_j \in \{\pm 1\}, j = 1, \cdots, n$. Define $d_i \triangleq \min\Big(\big|\bar{y}_i - \sum_{j=i}^n |r_{ij}|\big|, \big|\bar{y}_i + \sum_{j=i}^n |r_{ij}|\big|\Big)$ if $|\bar{y}_i| > \sum_{j=i}^n |r_{ij}|$, otherwise 0. Obviously $(\bar{y}_i - \sum_{j=i}^n r_{ij}x_j)^2 \geq d_i^2$. Then from (4) it follows that

$$(\bar{y}_m - \sum_{j=m}^n r_{mj}x_j)^2 \leq \beta^2 - \sum_{i=1}^{m-1} d_i^2. \tag{6}$$

Define $\rho \triangleq (\beta^2 - \sum_{i=1}^{m-1} d_i^2)^{1/2}$. Let $I^+ \triangleq \{j\ |r_{mj} \geq 0, m \leq j \leq n\}$ and $I^- \triangleq \{j\ |r_{mj} < 0, m \leq j \leq n\}$. Then (6) is equivalent to

$$\bar{y}_m - \rho \leq \sum_{j \in I^+} r_{mj}x_j + \sum_{j \in I^-} r_{mj}x_j \leq \bar{y}_m + \rho. \tag{7}$$

For each x_j, $m \leq j \leq n$, define the following bijective transformation (see [3]):

$$\bar{b}_j \triangleq \begin{cases} \frac{1+x_j}{2}, & j \in I^+ \\ \frac{1-x_j}{2}, & j \in I^- \end{cases} \tag{8}$$

so that $\bar{b}_j \in \{0, 1\}$. Define $\bar{a}_j \triangleq 2|r_{mj}|, j = m, \ldots, n$, then we obtain from (7) that

$$\bar{y}_m + \sum_{j=m}^n |r_{mj}| - \rho \leq \sum_{j=m}^n \bar{a}_j\bar{b}_j \leq \bar{y}_m + \sum_{j=m}^n |r_{mj}| + \rho. \tag{9}$$

We sort these nonnegative $\bar{a}_j$ in nondecreasing order:

$$0 \leq \bar{a}_{j_1} \leq \bar{a}_{j_2} \leq \cdots \leq \bar{a}_{j_N}.$$

The nonnegativity of $\bar{a}_j$, which is not true in Algorithm DV, is crucial for our new algorithm. To simplify notation, define

$$\eta_L \triangleq \bar{y}_m + \sum_{i=m}^n |r_{mj}| - \rho,\ \eta_U \triangleq \bar{y}_m + \sum_{i=m}^n |r_{mj}| + \rho,$$

$$\boldsymbol{a} \triangleq [a_1, \cdots, a_N]^T \triangleq [\bar{a}_{j_1}, \bar{a}_{j_2}, \cdots, \bar{a}_{j_N}]^T,$$

$$\boldsymbol{b} \triangleq [b_1, \cdots, b_N]^T \triangleq [\bar{b}_{j_1}, \bar{b}_{j_2}, \cdots, \bar{b}_{j_N}]^T,$$

then the inequality (9) can be rewritten as

$$\eta_L \leq \boldsymbol{a}^T\boldsymbol{b} \leq \eta_U. \tag{10}$$

For any $\boldsymbol{b}$ with $b_j \in \{0, 1\}$, we can check if it satisfies (10). If so, we then obtain the corresponding $\boldsymbol{x}^{(2)}$ through (8) and solve the sub-ILS problem in (3). But for efficiency we want to avoid enumerating all possible $\boldsymbol{b}$. In the following we show how to do it based on the idea of [2]. Let S be the set of all possible 2^N binary sequences for $\boldsymbol{b}$. For every subset B of S, one associates two quantities, lower bound $lb(B)$ and upper bound $ub(B)$, satisfying $lb(B) \leq \boldsymbol{a}^T\boldsymbol{b} \leq ub(B), \forall \boldsymbol{b} \in B$. Define the subsets of S as follows: $S_i \triangleq \{\boldsymbol{b} \in S | \omega(\boldsymbol{b}) = i\}, 0 \leq i \leq N$ where $\omega(\boldsymbol{b}) \triangleq \sum_{k=1}^N b_k$. Obviously these subsets S_i form a disjoint partition of S (to be called a depth-1 partition) and have the optimal lower bounds and upper bounds: $lb(S_0) = ub(S_0) = 0, lb(S_i) = \sum_{k=1}^i a_k, ub(S_i) = \sum_{k=N-i+1}^N a_k, 1 \leq i \leq N$. Since all $a_k \geq 0$ and are in nondecreasing order, it follows that

$$lb(S_0) \leq lb(S_1) \leq \cdots \leq lb(S_N)\ , \tag{11}$$

$$ub(S_0) \leq ub(S_1) \leq \cdots \leq ub(S_N). \tag{12}$$

These subsets starting with S_0, then S_1 until S_N are tested against (10). If $lb(S_i) > \eta_U$, then S_i and all the subsequent subsets after S_i do not need to be tested, since no any $\boldsymbol{b}$ in these subsets will satisfy (10). If $ub(S_i) < \eta_L$, only S_i can be discarded. If $lb(S_i) \leq \eta_U$ and $ub(S_i) \geq \eta_L$, we need to test all the elements of S_i against (10). Note that the cardinality of S_i is $\binom{N}{i}$ and it can be quite large when N is large. In order to discard as many sequences in S as possible before testing them against (10), we propose the following deepest depth partition strategy.

Define $S_{i,j} \triangleq \{\boldsymbol{b} \in S_i | b_j = 1 \text{ and } b_k = 0, 1 \leq k \leq j-1\}$ for $1 \leq i \leq N, 1 \leq j \leq N-i+1$. It is easy to observe that they form a disjoint partition (depth-2) of S_i and we have the optimal lower bound and upper bound for each $S_{i,j}$: $lb(S_{i,j}) = \sum_{k=j}^{i+j-1} a_k, ub(S_{i,j}) = a_j + \sum_{k=N-i+2}^N a_k$. Obviously $lb(S_{i,j})$ and $ub(S_{i,j})$ for $j = 1, \ldots, N-i+1$ are in nondecreasing order, respectively. Then we continue the above partition process for each subset, unless the subset has only one element or has been discarded. Finally each left subset has only one element.

Now we describe the whole process of our new GSD algorithm for solving (3). The algorithm searches the subsets S_i through the order: $S_0, S_1, S_2, \cdots, S_N$. First we find the largest index i_1 and the smallest index i_2 such that $ub(S_{i_1}) < \eta_L$ and $lb(S_{i_2}) > \eta_U$. Due to (11) and (12), the subsets $S_0, \cdots, S_{i_1}$ and $S_{i_2}, \cdots, S_N$ can be discarded. Next we consider the subsets from S_{i_1+1} to S_{i_2-1}. Define the set $\Gamma = \{S_{i_1+1}, \cdots, S_{i_2-1}\}$. Let the first subset of Γ be Γ_1, i.e., S_{i_1+1}. If Γ_1 contains only one element $\boldsymbol{b}$, we check if it satisfies (10) and if it does we solve the corresponding sub-ILS problem. Otherwise, we do the depth-2 partition to Γ_1 and deal with the elements of set Γ_1 in a recursive way. After Γ_1 is done, we delete it from Γ. In the left Γ, we delete each Γ_j such that $lb(\Gamma_j) > \eta_U$. Then we deal with the first subset in Γ again. The whole process is terminated when Γ is empty.

```
**Algorithm** Recursive GSD for problem (3)
Input: The trapezoidal matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^m$, the initial $\eta_L, \eta_U, \beta^2$, the set $S$.
Output: The optimal solution $\boldsymbol{x} \in \mathbb{Z}^n$ to (3) satisfying $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|_2^2 < \beta^2$.
function: $\boldsymbol{x} = \mathrm{UILS}(\boldsymbol{R}, \bar{\boldsymbol{y}}, \eta_L, \eta_U, \beta^2, S)$
if set $S$ contains only one element $\boldsymbol{b}$
    if $\boldsymbol{b}$ satisfies (10)
        Solve the corresponding sub-ILS problem.
        If a solution exits, let it be $\boldsymbol{x}$, and update
        $\eta_L, \eta_U, \beta^2$, otherwise set $\boldsymbol{x} =$ Null.
    end
else
    Partition $S$ into disjoint subsets $S_1, S_2, \cdots, S_{L_S}$,
     and define $\Gamma = \{S_1, S_2, \cdots, S_{L_S}\}$.
    Find the largest index $i_1$ such that $ub(S_{i_1}) < \eta_L$,
     and delete $S_1, \cdots, S_{i_1}$ from $\Gamma$.
    while $\Gamma \neq \emptyset$
        Find the smallest index $i_2$ such that
         $lb(S_{i_2}) > \eta_U$, delete $S_{i_2}, \cdots, S_{L_\Gamma}$ from $\Gamma$.
        if $\Gamma \neq \emptyset$
            Let $\Gamma_1$ be the first subset of $\Gamma$
            $\boldsymbol{x}' = \mathrm{UILS}(\boldsymbol{R}, \bar{\boldsymbol{y}}, \eta_L, \eta_U, \beta^2, \Gamma_1)$
            if $\boldsymbol{x}' \neq$ Null, set $\boldsymbol{x} = \boldsymbol{x}'$, end
            Delete $\Gamma_1$ from $\Gamma$
        end
    end
end
```

The above algorithm is more efficient in generating a sequence of sub-ILS problems than Algorithm DV given in [2]. The key is that we used the bijective transformation (8), which is an extension of that used in Algorithm DV, so that in depth-1 partition (say) $lb(\cdot)$ and $ub(\cdot)$ have the ideal orders (11) and (12), which make it possible to discard more sequences which do not satisfy (10) than Algorithm DV. The paper [2] says that high depth is necessary to effectively reduce the computational complexity for severely under-determined systems. But it does not show how to choose the depth of Algorithm DV. This leaves a user to decide the depth. However, there is no such problem with our algorithm, since it has the deepest depth, which allows us to make full use of the advantage of this partition technique.

2.2. Reduction from problem (1) to problem (3)

In solving a box constrained overdetermined ILS problem, to make the search process fast, the reduction process has to involve column-reordering of the channel matrix $\boldsymbol{A}$ (see, e.g., [6] and [7]). But for the UILS problem (1), to our knowledge, no GSD algorithm in the literature has considered to apply this strategy. In this section we propose a column-reordering strategy in the reduction from (1) to (3), i.e., we will find a good permutation matrix $\boldsymbol{P}$ in the QR decomposition: $\boldsymbol{AP} = \boldsymbol{QR}$. This strategy can significantly reduce the generation time of sub-ILS problems and decrease the number of these sub-ILS problems.

Suppose

$$i \triangleq \min\{k : lb(S_k) > \eta_U = \bar{y}_m + \sum_{j=m}^{n} |r_{m,j}| + \rho\} \quad (13)$$

then from (10) and (11) in Section 2.1 we see all the remaining subsets S_k for $k \geq i$ can be discarded without generating any sub-ILS problems. Therefore we would like i in (13) to be as close to 1 as possible. In addition to this, we also want to make η_U as small as possible. Its purpose is not just to decrease i in (13), but to help to reject $\boldsymbol{b}$ in other subsets S_j satisfying $lb(S_j) \leq \eta_U$ in generating a sub-ILS problem.

In the following we describe our reduction process. We first compute the QR decomposition of $\boldsymbol{A}$ by using the standard column pivoting strategy (see, e.g., [4, Sect 5.4.1]), which gives the QR decomposition of $\boldsymbol{A}\bar{\boldsymbol{P}}$, where $\bar{\boldsymbol{P}}$ is a permutation matrix. This will tend to make $|r_{m,j}|$ ($j = m, \dots, n$) (see (13)) small, since, roughly speaking, m larger columns of $\boldsymbol{A}$ (in terms of the 2-norm) have been moved to the front. Then we compute $\bar{\boldsymbol{y}}$. In order to make η_U small, we always keep $\bar{y}_m$ to be nonpositive (if $\bar{y}_m > 0$, we can simply multiply $\bar{y}_m$ and the last row of $\boldsymbol{R}$ by -1). Then we obtain the index i satisfying (13). Note that any reordering of the last $n - m + 1$ columns of $\boldsymbol{A}\bar{\boldsymbol{P}}$ will not change the value of i, since nothing in the inequality in (13) will change. The reordering of the first $m - 1$ columns of $\boldsymbol{A}\bar{\boldsymbol{P}}$ may only change ρ in (13) a little bit and is unlikely to change the smallest index i usually. So we only consider the following column permutations. For $j = 1, \dots, m-1$, we interchange the j-th column and m-th column of $\boldsymbol{R}$ (which is equivalent to the corresponding column interchange in $\boldsymbol{A}\bar{\boldsymbol{P}}$). After each interchange, we compute the QR decomposition of the permuted $\boldsymbol{R}$. This can be done in an efficient way by using the Givens rotations, since the permuted $\boldsymbol{R}$ has structure. In each time we have to simultaneously update $\bar{\boldsymbol{y}}$ by the same Givens rotations. Then we can obtain the corresponding i in (13). Finally we know which permutation gives the smallest i. If there is more than one permutation leading to the same smallest i, we choose the one which gives the smallest η_U. So finally the column-reordering of $\boldsymbol{A}$ is determined.

2.3. Determining the initial radius of the hyper-sphere

The radius β of the initial hyper-sphere in (4) is crucial to the cost of a GSD algorithm. If it is too small, the optimal solution might be outside the searched hyper-sphere and the GSD algorithm has to be restarted with a bigger β (in [2] and [3] β^2 is increased by a factor of 2). If it is too large, there are too many candidates inside the searched hyper-sphere. In [2] and [3], the initial β^2 is taken as $m/2$. It appears that this choice may still have the above problems.

In our new GSD algorithm, we obtain the initial β by solving the constrained *real* LS problem. Let $\boldsymbol{x}' = \frac{\boldsymbol{e}+\boldsymbol{x}}{2}$ where $\boldsymbol{e} = [1, 1, \cdots, 1]^T \in \mathbb{R}^n$. Then the problem (3) is equivalent to $\min_{\boldsymbol{x}' \in \{0,1\}^n} \|\boldsymbol{y}' - \boldsymbol{R}'\boldsymbol{x}'\|_2^2$, where $\boldsymbol{y}' = \bar{\boldsymbol{y}} + \boldsymbol{R}\boldsymbol{e},\ \boldsymbol{R}' = 2\boldsymbol{R}$. We solve the *real* LS problem $\min_{\boldsymbol{0} \le \boldsymbol{x}' \le \boldsymbol{e}} \|\boldsymbol{y}' - \boldsymbol{R}'\boldsymbol{x}'\|_2^2$ to obtain the real solution $\boldsymbol{x}'$ by a gradient projection method (see, e.g, [8, Sec 16.6]). Then we take $\beta^2 = \|\boldsymbol{y}' - \boldsymbol{R}'\lfloor \boldsymbol{x}' \rceil\|_2^2$, where the i-th entry of $\lfloor \boldsymbol{x}' \rceil$ is the nearest integer to x_i'.

3. SIMULATIONS

In this section, we compare the computational performance of our new recursive GSD algorithm given in Section 2, to be referred to as Algorithm CY for consistence and convenience, with Algorithm DV and Algorithm YLH. All our computations were performed in MATLAB 7.0.

The elements of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ were drawn from an i.i.d. zero-mean, unit variance Gaussian distribution. The input vector $\boldsymbol{y} \in \mathbb{R}^m$ was constructed as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v}$ where each entry of $\boldsymbol{x}$ was taken from $\{\pm 1\}$, the elements of the noise vector $\boldsymbol{v} \in \mathbb{R}^m$ were drawn from an i.i.d zero-mean Gaussian distribution. We used the same conventional sphere decoding algorithm for solving each sub-ILS problem in the three algorithms, which was the Schnorr-Euchner based search algorithm with the V-BLAST preprocessing strategy incorporated (see [7]). In Algorithm DV, we took the depth to be $n-m-1$. The cost of the different GSD algorithms is measured by the total number of flops. The cost of the column-reordering and the computation of the initial radius of the hyper-sphere in Algorithm CY is not considered, since it is negligible compared with the other cost. For each case, we performed 100 runs and took the average number of flops.

For $\boldsymbol{v} \sim N(0, 0.1^2\boldsymbol{I}_m)$, $m = 10, n-m = 3, 4, \ldots 14$, we give the average number of flops of the three different GSD algorithms in Fig. 1. From this figure, we observe that our new GSD algorithm becomes more and more efficient than Algorithm DV and Algorithm YLH when $n - m$ increases. When $n - m = 14$, the cost of either Algorithm DV or Algorithm YLH is 20 times more than that of our Algorithm CY.

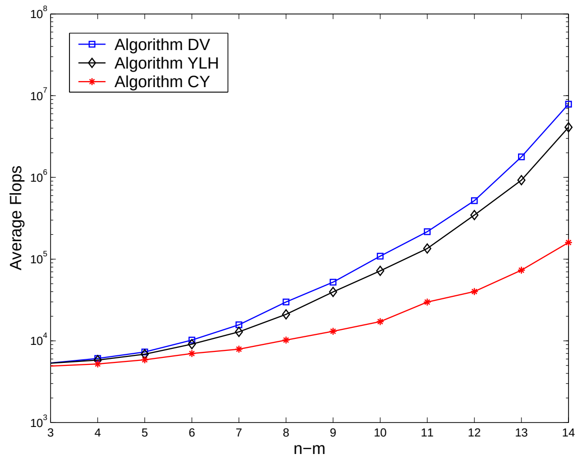

Fig. 1. Average flops vs n-m, $\boldsymbol{v} \sim N(\boldsymbol{0}, 0.1^2\boldsymbol{I})$.

4. SUMMARY

In this paper we proposed a recursive GSD algorithm for optimal decoding of under-determined MIMO systems. We showed how to efficiently generate the sequence of sub-ILS problems by modifying the set partition technique of [2]. We also presented a reduction process with column-reordering strategy, which can significantly reduce the computational cost. Moreover, we gave a method to determine a good initial radius of the hyper-sphere to overcome the difficulty with current algorithms. Simulation results showed that our new GSD algorithm is much more efficient than those given in [2] and [3].

5. REFERENCES

[1] M.O. Damen, K. Abed-Meraim, and J.C. Belfiore, “Generalized sphere decoder for asymmetrical space-time communication architecture,” *IEEE Electronics Letters*, vol. 36, pp. 166–167, 2000.

[2] P. Dayal and M.K. Varanasi, “A fast generalized sphere decoder for optimum decoding of under-determined MIMO systems,” in *41st Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003, pp. 1216–1225.

[3] Z. Yang, C. Liu, and J. He, “A new approach for fast generalized sphere decoding in MIMO systems,” *IEEE Sig. Proc. Letters*, vol. 12, no. 1, pp. 41–44, 2005.

[4] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.

[5] E. Viterbo and J. Boutros, “A universal lattice code decoder for fading channels,” *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[6] X.-W. Chang and Q. Han, “Solving box-constrained integer least-squares problems,” *submitted*, 2005.

[7] M.O. 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.

[8] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.