FURTHER RESULTS ON SPEEDING UP THE SPHERE DECODER

M. Stojnic, H. Vikalo, and B. Hassibi

California Institute of Technology, Pasadena, CA
e-mail: mihailo,hvikalo,hassibi@systems.caltech.edu

ABSTRACT

In many communication applications, maximum-likelihood decoding reduces to solving an integer least-squares problem which is NP hard in the worst-case. On the other hand, it has recently been shown that, over a wide range of dimensions and SNR, the sphere decoder can be used to find the exact solution with an expected complexity that is roughly cubic in the dimension of the problem. However, the computational complexity becomes prohibitive if the SNR is too low and/or if the dimension of the problem is too large. In earlier work, we targeted these two regimes attempting to find faster algorithms by pruning the search tree beyond what is done in the standard sphere decoder. The search tree is pruned by computing lower bounds on the possible optimal solution as we proceed to go down the tree. A trade-off between the computational complexity required to compute the lower bound and the size of the pruned tree is readily observed: the more effort we spend in computing a tight lower bound, the more branches that can be eliminated in the tree. Thus, even though it is possible to prune the search tree (and hence the number of points visited) by several orders of magnitude, this may be offset by the computations required to perform the pruning. In this paper, we propose a computationally efficient lower bound which requires solving a single semi-definite program (SDP) at the top of the search tree; the solution to the SDP is then used to deduce the lower bounds on the optimal solution on all levels of the search tree. Simulation results indicate significant improvement in the computational complexity of the proposed algorithm over the standard sphere decoding.

1. INTRODUCTION

We are interested in finding the exact solution to the following problem,

$$\min_{\mathbf{s}\in\mathcal{D}\subset\mathcal{Z}^m} \|\mathbf{x}-H\mathbf{s}\|_2, \qquad (1)$$

where $\mathbf{x}\in\mathcal{R}^m$, $H\in\mathcal{R}^{m\times m}$ and $\mathcal{D}$ refers to some subset of the integer lattice $\mathcal{Z}^m$; in this paper, we focus on $\mathcal{D} = \{-\frac{1}{2},\frac{1}{2}\}^m$. The main idea of the sphere decoder algorithm [1] for solving (1) is based on finding all points $\mathbf{s}$ such that $\|\mathbf{x}-H\mathbf{s}\|_2$ lies within some adequately chosen radius d, i.e., on finding all $\mathbf{s}$ such that

$$d^2 \geq \|\mathbf{x}-H\mathbf{s}\|_2^2, \qquad (2)$$

and then choosing the one that minimizes the objective function. Using the QR-decomposition $H=QR$, with Q unitary and R upper triangular, we can reformulate (2) as

$$d^2 \geq \|\mathbf{y}-R\mathbf{s}\|_2^2, \qquad (3)$$

where we have defined $\mathbf{y}=Q^*\mathbf{x}$. Since R is upper-triangular, (3) can be further rewritten as

$$d^2 \geq \|\mathbf{y}_{k:m}-R_{k:m,k:m}\mathbf{s}_{k:m}\|^2 + \|\mathbf{y}_{1:k-1}-R_{1:k-1,1:k-1}\mathbf{s}_{1:k-1}-R_{1:k-1,k:m}\mathbf{s}_{k:m}\|^2, \qquad (4)$$

for any $2\leq k\leq m$, where the subscripts determine the entries the various vectors and matrices run over. A necessary condition for (3) can therefore be obtained by omitting the second term on the RHS of the above expression to yield

$$d^2 \geq \|\mathbf{y}_{k:m}-R_{k:m,k:m}\mathbf{s}_{k:m}\|^2. \qquad (5)$$

The sphere decoder finds all points $\mathbf{s}$ in (2) by proceeding inductively on (5), starting from $k=m$ and proceeding to $k=1$. In other words for $k=m$, it determines all one-dimensional lattice points $\mathbf{s}_m$ such that $d^2\geq(\mathbf{y}_m-R_{m,m}\mathbf{s}_m)^2$, and then for each such one-dimensional lattice point $\mathbf{s}_m$ determines all possible values for $\mathbf{s}_{m-1}$ such that

$$\begin{aligned} d^2 &\geq \|\mathbf{y}_{m-1:m}-R_{m-1:m,m-1:m}\mathbf{s}_{m-1:m}\|^2 \\ &= (\mathbf{y}_m-R_{m,m}\mathbf{s}_m)^2 \\ &+ (\mathbf{y}_{m-1}-R_{m-1,m-1}\mathbf{s}_{m-1}-R_{m-1,m}\mathbf{s}_m)^2. \end{aligned}$$

This gives all possible two-dimensional lattice points; one then proceeds in a similar fashion until $k=1$. The sphere decoder thus generates a tree, where the branches at the $(m-k+1)$th level of the tree correspond to all $m-k+1$-dimensional lattice points satisfying (5). In this manner at the bottom of the tree (the m-th level) all points satisfying (2) are found.

This work was supported in part by the National Science Foundation under grant no. CCR-0133818 and CCR-0326554, by the David and Lucille Packard Foundation, and by Caltech's Lee Center for Advanced Networking.

The computational complexity of the sphere decoder depends on how d is chosen. In communications we usually can assume

$$\mathbf{x} = H\mathbf{s} + \mathbf{w}, \tag{6}$$

where the entries of $\mathbf{w}$ are independent $\mathcal{N}(0, \sigma^2)$ random variables. In [2] it is shown that, if the radius is chosen appropriately based on the statistical characteristics of the noise $\mathbf{w}$, then over a wide range of SNRs and problem dimensions the expected complexity of the sphere decoder is roughly cubic.

2. SPEEDING UP THE SPHERE DECODER

The above assertion unfortunately fails and the computational complexity becomes increasingly prohibitive if the SNR is too low and/or if the dimension of the problem is too large (see [9]). Increasing the dimension of the problem clearly is useful. Moreover, the use of the sphere decoder in low SNR situations is also important when one is interested in obtaining soft information to pass onto an iterative decoder (see, e.g., [3, 4]). To reduce the computational complexity one approach is to resort to suboptimal methods based either on heuristics (see, e.g., [5]) or some form of statistical pruning (see [6]). The other approach is to use different implementations of the original sphere decoder (see e.g. [8] and references therein).

In [7], we attempted to reduce the computational complexity of the sphere decoder while still finding the *exact* solution. Let us recall how this may be done. Clearly, the complexity of the algorithm depends on the size of the search tree since each branch in the tree is visited and appropriate computations are then performed. Thus, one approach would be to reduce the size of the tree beyond that which is suggested by (5). To do so, suppose that we had some way of computing a lower bound on the optimal value of the second term of the RHS of (4):

$$LB^{(k-1)} = LB(\mathbf{y}_{1:k-1}, R_{1:k-1,1:m}, \mathbf{s}_{k:m}) \leq \min_{\mathbf{s}_{1:k-1} \in \mathcal{D} \subset \mathcal{Z}^{k-1}} \|\mathbf{y}_{1:k-1} - R_{1:k-1,1:k-1}\mathbf{s}_{1:k-1} - R_{1:k-1,k:m}\mathbf{s}_{k:m}\|^2,$$

where we have emphasized the fact that the lower bound is a function of $\mathbf{y}_{1:k-1}$, $R_{1:k-1,1:m}$, and $\mathbf{s}_{k:m}$. Provided our lower bound is nontrivial, i.e., $LB^{(k-1)} > 0$, we can replace (5) by [1]

$$d^2 - LB^{(k-1)} \geq \|\mathbf{y}_{k:m} - R_{k:m,k:m}\mathbf{s}_{k:m}\|^2. \tag{7}$$

This is certainly a more restricted condition than (5) and so will lead to the elimination of more points from the tree. Note that (7) will not result in missing any lattice points from (2) since we have used a lower bound for the remainder of the cost in (4). Assuming that we have some way of computing a lower bound, we state the modification of the standard sphere decoder algorithm based on the use of (7) with $LB^{(k-1)} > 0$. The algorithm uses Schnorr-Euchner strategy with the radius update.

Input: Q, R, x, $y = Q^*x$, $d = \hat{d}$, $ll_{1:m} = 0_{1\times m}$.

1. Set $k = m$, $d_m^2 = d^2$, $y_{m|m+1} = y_m$
2. (Bounds for s_k) Set $ub(s_k) = \lfloor \frac{\sqrt{d_k^2 - (d^2 - \hat{d}^2)} + y_{k|k+1}}{r_{k,k}} \rfloor$, $lb(s_k) = \lceil \frac{-\sqrt{d_k^2 - (d^2 - \hat{d}^2)} + y_{k|k+1}}{r_{k,k}} \rceil$, $l_k = \lfloor \frac{lb(s_k) + ub(s_k) + 1}{2} \rfloor$, $u_k = l_k + 1$
3. (Zig-zag through s_k)
   if $ll_k = 0$, $s_k = l_k$, $l_k = l_k - 1$, $ll_k = 1$, otherwise $s_k = u_k$, $u_k = u_k + 1$, $ll_k = 0$.
   If $lb(s_k) \leq s_k \leq ub(s_k)$, go to 4, else go to 5.
4. if $LB^{(k-1)} + (y_{k|k+1} - r_{k,k}s_k)^2 - d_k^2 + (d^2 - \hat{d}^2) > 0$, go to 3, else go to 6.
5. (Increase k) $k = k + 1$; if $k = m + 1$ terminate algorithm, else go to 3.
6. (Decrease k) If $k = 1$ go to 7. Else $k = k - 1$, $y_{k|k+1} = y_k - \sum_{j=k+1}^{m} r_{k,j}s_j$, $d_k^2 = d_{k+1}^2 - (y_{k+1|k+2} - r_{k+1,k+1}s_{k+1})^2$, and go to 2.
7. Solution found. Save s and its distance from x, $\hat{d} = d_m^2 - d_1^2 + (y_1 - r_{1,1}s_1)^2$, and go to 3.

Clearly, the tighter the lower bound $LB^{(k-1)}$, the more points that will be pruned from the tree. Of course, we cannot hope to find the optimal lower bound since this requires solving an integer least-squares problem (which was our original problem to begin with). Instead, we consider obtaining lower bounds on the integer least-squares problem

$$\min_{\mathbf{s}_{1:k-1} \in \mathcal{D} \subset \mathcal{Z}^{k-1}} \|\mathbf{z}_{(k-1)} - R_{1:k-1,1:k-1}\mathbf{s}_{1:k-1}\|^2, \tag{8}$$

where we have defined $\mathbf{z}_{(k-1)} = \mathbf{y}_{1:k-1} - R_{1:k-1,k:m}\mathbf{s}_{k:m}$.

Note that finding a lower bound on (8) requires *some* computational effort. Therefore, it is a natural question to ask whether the benefits of additional pruning outweigh the additional complexity incurred by computing a lower bound. In [7], we considered a duality based lower bound obtained by solving

$$\begin{aligned} LB_{SDP}^{(k-1)} = \max \quad & \mathrm{Tr}(\Lambda) \\ \text{subject to} \quad & Q_{k-1} \succeq \Lambda, \quad \Lambda \text{ is diagonal}, \end{aligned} \tag{9}$$

where

$$Q_{k-1} = \begin{bmatrix} \frac{1}{4}R_{1:k-1,1:k-1}^T R_{1:k-1,1:k-1} & -\frac{1}{2}R_{1:k-1,1:k-1}^T \mathbf{z}_{(k-1)} \\ -\frac{1}{2}\mathbf{z}_{(k-1)}^T R_{1:k-1,1:k-1} & \mathbf{z}_{(k-1)}^T \mathbf{z}_{(k-1)} \end{bmatrix}.$$

[1] $LB^{(k-1)} = 0$, of course, simply corresponds to the standard sphere decoder.

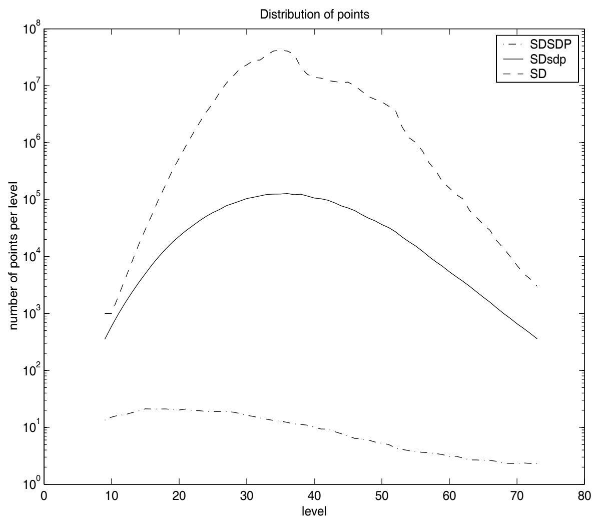

Fig. 1. Comparison of number of points per level in search tree, $m = 100$, SNR = 10dB, $\mathcal{D} = \{-\frac{1}{2}, \frac{1}{2}\}^{k-1}$

The dashed and the solid lines in Figure 1 compare the average number of points on each level of the search tree visited by the basic sphere decoding algorithm with the corresponding number of points visited by the sphere decoding algorithm which employs a lower bound (9). We refer to the former as the SD-algorithm and to the latter as the SDSDP-algorithm. As evident from Figure 1, the number of points in the search tree visited by the SDSDP-algorithm is several orders of magnitude smaller than that visited by the SD-algorithm. Therefore, a good lower bound can help prune the tree much more efficiently than the standard sphere decoding alone. However, computing LB_{SDP} requires solving an SDP per each point in the search tree. Since this requires computational effort roughly cubic in k, the total flop count savings are not as significant as the savings in the number of examined tree points shown in Figure 2. Therefore, there is merit in searching for lower bounds that may not be as tight as (9), but which require significantly lower computational effort.

Such a lower bound is derived in the next section. As a preview of the results to come, the number of points per level in the search tree where this new bound is used is shown in Figure 1. Clearly, the new bound significantly reduces the number of points visited by the basic sphere decoding algorithm. Moreover, it turns out that its computation requires very low additional effort beyond the basic sphere decoding complexity, as we show in the next section.

3. BOUND ON SOLUTION OF SDP

In this section, we derive a lower bound $LB_{sdp}^{(k-1)}$ on the value of $LB_{SDP}^{(k-1)}$ in (9). To this end, let $\hat{\Lambda}$ denote the optimal solution of

$$\begin{array}{ll} \max & \mathrm{Tr}(\Lambda) \\ \text{subject to} & Q \succeq \Lambda, \quad \Lambda \text{ is diagonal}, \end{array} \tag{10}$$

where

$$Q = \begin{bmatrix} \frac{1}{4}R^T R & -\frac{1}{2}R^T \mathbf{y} \\ -\frac{1}{2}\mathbf{y}^T R & \mathbf{y}^T \mathbf{y} \end{bmatrix},$$

and let $LL^T = \frac{1}{4}R^T R - \hat{\Lambda}$, where L is a lower triangular matrix. Also, let $M = L^{-1}R^T$. Using the fact that the matrices L and R^T are lower triangular we obtain

$$\begin{aligned} (L_{1:k-1,1:k-1})^{-1} &= (L^{-1})_{1:k-1,1:k-1}, \\ L_{1:k-1,1:k-1}(L_{1:k-1,1:k-1})^T &= \frac{1}{4}R^T_{1:k-1,1:k-1}R_{1:k-1,1:k-1} \\ &- \hat{\Lambda}_{1:k-1,1:k-1}, \end{aligned}$$

and $M_{1:k-1,1:k-1} = (L_{1:k-1,1:k-1})^{-1}R^T_{1:k-1,1:k-1}$. Furthermore, let

$$\lambda_k = \mathbf{z}^T_{(k-1)}\mathbf{z}_{(k-1)} - \frac{1}{4}\mathbf{z}^T_{(k-1)}M^T_{1:k-1,1:k-1}M_{1:k-1,1:k-1}\mathbf{z}_{(k-1)} \tag{11}$$

and let

$$LB_{sdp}^{(k-1)} = \begin{cases} \sum_{i=1}^{k-1} \hat{\Lambda}_{i,i} + \lambda_k & \text{if} \quad \lambda_k \geq 0 \\ 0 & \text{if} \quad \lambda_k < 0. \end{cases} \tag{12}$$

Now it is clear that $LB_{sdp}^{(k-1)} \leq LB_{SDP}^{(k-1)}$ since

$$LB_{sdp}^{(k-1)} = \mathrm{Tr}(\mathrm{diag}(\hat{\Lambda}_{1,1}, \hat{\Lambda}_{2,2}, \ldots, \hat{\Lambda}_{k-1,k-1}, \lambda_k))$$

and $\mathrm{diag}(\hat{\Lambda}_{1,1}, \hat{\Lambda}_{2,2}, \ldots, \hat{\Lambda}_{k-1,k-1}, \lambda_k)$ is an admissible matrix in (9) if $\lambda_k \geq 0$. On the other hand, if $\lambda_k < 0$, $LB_{sdp}^{(k-1)} = 0$ and clearly $LB_{sdp}^{(k-1)} \leq LB_{SDP}$.

We refer to the algorithm which uses $LB_{sdp}^{(k-1)}$ in place of $LB^{(k-1)}$ in (7) as the SDsdp-algorithm. Clearly, using $LB_{sdp}^{(k-1)}$ instead of $LB_{SDP}^{(k-1)}$ results in pruning fewer points in the search tree. However, computation of $LB_{sdp}^{(k-1)}$ is quite more efficient than the cubic computation of $LB_{SDP}^{(k-1)}$. In particular, unlike in the SDSDP-algorithm, we need to solve only one SDP – the one given by (10). Then we may compute $LB_{sdp}^{(k-2)}$ recursively from $LB_{sdp}^{(k-1)}$, which requires complexity linear in k. This is shown next.

Recall that $\mathbf{z}_{(k-1)} = \mathbf{y}_{1:k-1} - R_{1:k-1,k:m}\mathbf{s}_{k:m}$. It is easy to see that we can compute $\mathbf{z}_{(k-2)}$ from $\mathbf{z}_{(k-1)}$ as

$$\mathbf{z}_{(k-2)} = (\mathbf{z}_{(k-1)})_{1:k-2} - R_{1:k-2,k-1}s_{k-1}. \tag{13}$$

Furthermore, note that $\mathbf{b}^{(k-1)} = M_{1:k-1,1:k-1}\mathbf{z}_{(k-1)}$ can be computed recursively as

$$\begin{aligned} \mathbf{b}^{(k-2)} &= M_{1:k-2,1:k-2}\mathbf{z}_{(k-2)} \\ &= M_{1:k-2,1:k-2}((\mathbf{z}_{(k-1)})_{1:k-2} - R_{1:k-2,k-1}s_{k-1}) \\ &= M_{1:k-2,1:k-2}(\mathbf{z}_{(k-1)})_{1:k-2} - M_{1:k-2,1:k-2}R_{1:k-2,k-1}s_{k-1} \\ &= \mathbf{b}^{(k-1)}_{1:k-2} - (MR)_{1:k-2,k-1}s_{k-1}. \end{aligned} \tag{14}$$

Using $\mathbf{b}^{(k-2)}$ and $\mathbf{z}_{(k-2)}$ we compute λ_{k-1} from (11), and $LB_{sdp}^{(k-2)}$ from (12). The number of operations required to compute $LB_{sdp}^{(k-2)}$ in each node at the $(m-(k-2))$th level of the search tree is $4(k-2)$ additions and $2(k-2)$ multiplications. For the basic sphere decoder, the number of operations per each node at the $(m-k+1)$th level is $(2k+17)$. This essentially means that the SDsdp algorithm performs about four times more operations per each node of the tree than the standard sphere decoder algorithm does. In other words, if the SDsdp-algorithm prunes at least four times more points that the basic sphere decoder, the new algorithm is faster in terms of the flop count. This is clearly the case in Figure 1, where the total number of the tree points visited by the SDsdp algorithm is 100-times smaller than the number of the tree points visited by the basic sphere decoder.

4. SIMULATION RESULTS

Figure 2 compares the expected complexity of the SDsdp-algorithm to the expected complexity of the SD-algorithm. The plots on Figure 2 were generated for the system of dimension $m = 50$. Note that the signal-to-noise ratio in Figure 2 is defined as SNR $= 10\log_{10}\frac{m}{4\sigma^2}$, where σ^2 is the variance of each component of the noise vector $\mathbf{w}$. Both algorithms choose the initial search radius statistically as in [2], and update the radius every time the bottom of the tree is reached. As the simulation results in Figure 2 indicate, the SDsdp algorithm runs more than 10 times faster than the SD algorithm. [Needless to say, the bit error-rate performance of both algorithms coincide with the maximum-likelihood.]

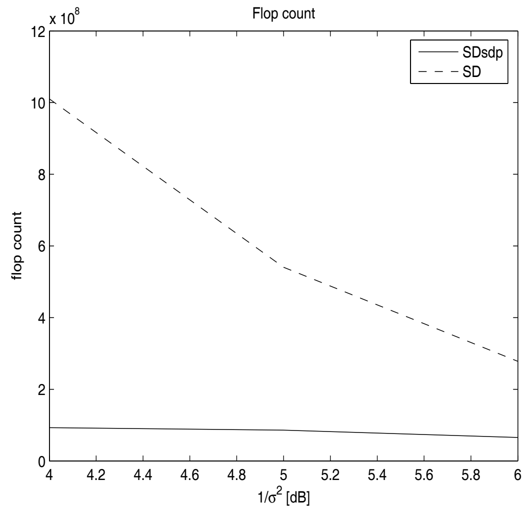

Fig. 2. Comparison of flop count, $m = 50$, $\mathcal{D} = \{-\frac{1}{2}, \frac{1}{2}\}^{k-1}$

5. SUMMARY AND DISCUSSION

Adding a lower bound on the remainder of the cost function has the potential to prune the search tree significantly more than the standard sphere decoder algorithm prunes. However, more significant pruning of the search tree does not, in general, guarantee that the modified algorithm will perform faster than the standard sphere decoder algorithm. This is due to the additional computations required by the modified algorithm to find a lower bound in each node of the search tree. Therefore, the lower bound on one hand has to be as tight as possible in order to prune the search tree as much as possible, and on the other hand it should be efficiently computable.

Led by these two main characteristics, in this paper we introduced a new type of a lower bound. The new bound is a lower bound on the solution of the SDP-relaxation of the $(0, 1)$–integer least-squares problem. It requires solving only one m-dimensional SDP, where m is the dimension of the problem, whose solution is then efficiently processed to obtain bounds at each tree level k; these additional operations for computing the lower bounds are only linear in k. Simulation results show that the new SDsdp algorithm outperforms the basic sphere decoder algorithm in terms of the flop count.

The main computational burden of the SDsdp algorithm is in solving the m-dimensional SDP. Seeking efficient ways of obtaining the solution to the SDP is of interest for future work.

6. REFERENCES

[1] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, pp. 463-471, April 1985.

[2] B. Hassibi, and H. Vikalo, "On the sphere decoding algorithm. Part I: The expected complexity," *IEEE Trans. on Signal Processing*, August 2005..

[3] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel", *IEEE Trans. on Comm.*, March, 2003.

[4] H. Vikalo, B. Hassibi and T. Kailath, "Iterative decoding for MIMO channels via modified sphere decoder", *IEEE Trans. on Wireless Comm.*, November, 2004.

[5] H. Artes, D. Seethaler and F. Hlawatsch, "Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection," *IEEE Trans. on Signal Processing*, November, 2003.

[6] R. Gowaikar and B. Hassibi, "Efficient maximum-likelihood decoding via statistical pruning", submitted to *IEEE Trans. on Information Theory*.

[7] M. Stojnic, H. Vikalo, B. Hassibi, "A branch and bound approach to speed up the sphere decoder", *ICASSP*, March 2005

[8] K. Su, I. J. Wassel, "A new ordering for efficient sphere decoder", *in Proc. ICC*, May 2005

[9] J. Jalden, B. Ottersten, "On the complexity of sphere decoding in digital communications", *IEEE Trans. on Signal Processing*, April 2005