The Sorted-QR Chase Detector for Multiple-Input Multiple-Output Channels

Deric W. Waters and John R. Barry

School of ECE
Georgia Institute of Technology
Atlanta, GA 30332-0250 USA
{deric, barry}@ece.gatech.edu

Abstract — **The performance of a decision-feedback detector on a fading multiple-input multiple-output channel is limited by the low diversity of the first symbol detected. We propose a new family of detection techniques which overcomes this bottleneck by using a *list* detector for the first symbol; the list detector is then combined with a parallel bank of decision-feedback detectors, one for each element of the list. The detector family is parameterized by the length of the list, which can be adjusted to achieve a wide range of attractive trade-offs between performance and complexity. For example, on a 4-input 4-output Rayleigh-fading channel with uncoded 16-QAM inputs, one version of the proposed detector outperforms the popular minimum-mean-squared-error BLAST-ordered decision-feedback detector by 1.5 dB, while simultaneously requiring 6% fewer computations.**

I. Introduction

The promise of both high spectral efficiency and diversity to fading has led to a flurry of recent research in multiple-input multiple-output (MIMO) communication systems. A practical obstacle to the realization of a MIMO system is the complexity of MIMO detection. Specifically, the complexity of the maximum-likelihood (ML) detector [1], which minimizes word-error probability, grows exponentially with the number of channel inputs.

The BLAST-ordered decision-feedback (BDF) detector (also known as a successive interference canceller [2]) is a popular reduced-complexity detector for MIMO channels. It can approach the performance of the ML detector when there are many more channel outputs than inputs [2], but otherwise the BDF detector is significantly inferior to the ML detector.

The large gap in both performance and complexity between the ML and BDF detectors has motivated the search for alternative detection algorithms. Various combinations of the ML and BDF detectors have been proposed that improve on the performance of the BDF detector at the cost of increased complexity [3, 4]. The minimum mean-squared-error (MMSE) sphere detector of [5] and the MMSE-BDF detector combined with lattice reduction of [6] both achieve near ML performance while reducing the average complexity. The B-Chase detector of [7] implements multiple BDF detectors in parallel, and can approach ML performance with even lower complexity. The L-Chase detector of [7] further decreases complexity by implementing multiple linear detectors in parallel. The L-Chase and B-Chase detectors are so-named because of their close ties to the well-known Chase algorithm for soft decoding of error-control codes [8].

In this paper, we propose the *S-Chase* detector that implements multiple suboptimally ordered decision-feedback (DF) detectors in parallel. A key problem for all DF detectors on fading channels is the minimal diversity gain for the first symbol detected; this leads to a large probability of error that can dominate the overall error rate. The proposed S-Chase detector overcomes this bottleneck by considering multiple possibilities for the first symbol, running a separate DF detector for each possibility, and choosing the best of the resulting candidate hard decision vectors.

A key benefit of considering multiple possibilities for the first symbol is that it drastically reduces the importance of optimizing the detection ordering. Indeed, we will see that a easily computed but suboptimal ordering is sufficient to achieve good performance. This makes the overall complexity of the S-Chase detector very low, despite the bank of DF detectors. The suboptimal ordering used by the S-Chase detector is based on the sorted-QR decomposition of [9]. We will see that this S-Chase detector has an improved performance-complexity trade-off relative to the B-Chase [7] and lattice-reduced MMSE BDF (LR-BDF) [6] detectors.

Section II introduces the MIMO channel model, describes the zero-forcing and MMSE S-Chase detectors, and describes a computationally efficient implementation. Section III quantifies both the performance and complexity of the new detector. Finally, in Section IV we make concluding remarks.

This research was supported in part by National Science Foundation grants CCR-0082329 and CCR-0121565.

II. The S-Chase Detector

This paper considers a memoryless channel with N inputs $\boldsymbol{a} = [a_1, \ldots a_N]^T$ and M outputs $\boldsymbol{r} = [r_1, \ldots r_M]^T$:

$$\boldsymbol{r} = \mathbf{H}\boldsymbol{a} + \boldsymbol{n}\,, \tag{1}$$

where $\mathbf{H} = [\boldsymbol{h}_1, \ldots \boldsymbol{h}_N]$ is a complex $M \times N$ channel matrix whose i-th column is $\boldsymbol{h}_i$, and where $\boldsymbol{n} = [n_1, \ldots n_M]^T$ is additive white Gaussian noise. We assume that the columns of $\mathbf{H}$ are linearly independent, which implies that there are at least as many outputs as inputs, $M \geq N$. We assume that the noise components are uncorrelated circularly symmetric Gaussian random variables with complex variance σ^2, so that $E[\,\boldsymbol{n}\boldsymbol{n}^*] = \sigma^2\mathbf{I}$, where $\boldsymbol{n}^*$ denotes the conjugate transpose of $\boldsymbol{n}$. Further, we assume that the inputs are chosen from the same unit-energy alphabet A, and are uncorrelated, so that $E[\boldsymbol{a}\boldsymbol{a}^*] = \mathbf{I}$.

The S-Chase detector is basically an efficient way to implement multiple DF detectors in parallel. It can be broken down into five steps:

Step 1. Identify the index i of the first symbol detected.

Step 2. Calculate a list of the q "most likely" tentative decisions for the i-th symbol.

Step 3. Implement q DF detectors in parallel, one for each element of the initialization list.

Step 4. Of the q decision vectors, pick the "best."

The S-Chase detector is parameterized by the list length q. When the list length is unity ($q = 1$), the S-Chase detector reduces to a conventional DF detector with a suboptimal ordering based on the sorted-QR decomposition [9]. Fig. 1 shows a block diagram of steps 2 through 4. In the following we provide the details of each of the five steps.

Step 1. The S-Chase detector breaks down the ordering problem into two smaller problems: selecting the index i of the first symbol to detect, and choosing the order of the remaining symbols.

Consider first the problem of choosing i. The solution is well-known when $q = 1$, since in this case the S-Chase detector reduces to a DF detector. Therefore, a good choice for the first symbol would be the one with the *largest* post-detection SNR, as found using the BLAST ordering. On the other hand, when q is large it is better to choose the symbol with the *smallest* SNR [7]. One could formulate an optimal method for selecting i for a given q [7]. However, to keep complexity at a minimum, we propose instead a simple and low-complexity heuristic that does not require evaluation of the post-detection SNR values: *choose the index of the column of* $\mathbf{H}$ *with either the minimum or maximum norm, depending on the list length* q. This selection strategy can be summarized succinctly as follows:

$$i = \underset{j \in \{1, \ldots, N\}}{\arg\max} \; \| \boldsymbol{h}_j \|^m\,, \tag{2}$$

where $m = -1$ when $q > 3|A|/4$, and $m = 1$ when $q \leq 3|A|/4$.

Once the first symbol to detect has been selected, we propose that the order of the remaining symbols be defined by the following sorted-QR decomposition [9]:

$$\mathbf{H}\Pi = \mathbf{Q}\mathbf{G}\,, \tag{3}$$

where the $N \times N$ permutation matrix Π represents the symbol ordering, $\mathbf{Q} = [\boldsymbol{q}_1, \ldots \boldsymbol{q}_N]$ is an $M \times N$ matrix with orthonormal columns, and $\mathbf{G} = [\boldsymbol{g}_1, \ldots \boldsymbol{g}_N]$ is an $N \times N$ lower triangular matrix with real and positive diagonal elements $\{g_{1,1}, \ldots g_{N,N}\}$. To preserve the selection made by (2), the first column of Π is the i-th column of the identity matrix. The final $N-1$ columns of Π are chosen according to the sorted-QR decomposition [9], which attempts to place weaker symbols later in the detection order. Implementing the selection heuristic (2) requires no additional floating-point operations because the QR decomposition already requires the column norms of $\mathbf{H}$. Fig. 2 gives the pseudocode of a computationally efficient algorithm that finds Π, $\mathbf{Q}$, and $\mathbf{G}$. Fig. 2 also notes the number of floating-point operations required by each line.

Step 2. The S-Chase detector applies the whitened-matched filter to the observation, producing $\boldsymbol{z} = \mathbf{Q}^*\boldsymbol{r}$, which reduces to:

$$\begin{aligned} \boldsymbol{z} &= \mathbf{G}\tilde{\boldsymbol{a}} + \boldsymbol{v} \\ &= \textstyle\sum_k \tilde{a}_k \boldsymbol{g}_k + \boldsymbol{v}\,, \end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{a}} = \Pi^*\boldsymbol{a}$ is a permuted version of the transmitted symbol vector. The traditional DF detector would make the symbol nearest $z_1/g_{1,1}$ its first decision. The S-Chase detector generalizes this idea by producing a *list* of tentative symbols $\{s_1, \ldots s_q\}$, defined as the q symbols nearest to $z_1/g_{1,1}$.

Step 3. Given the decision list from step 2, The S-Chase detector next implements a bank of q DF detectors in parallel, one for each element of the list, with the first decision of each DF detector hard-wired to the corresponding element of the list $\{s_1, \ldots, s_q\}$. Specifically, the l-th DF detector is succinctly described by:

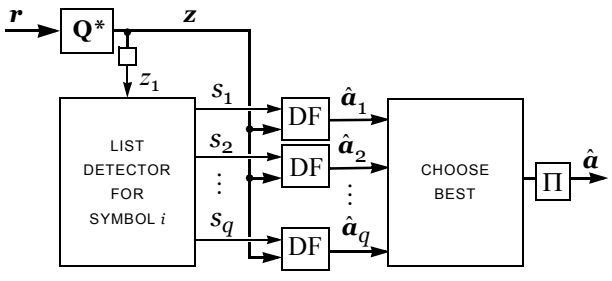

Fig. 1. Block diagram of the S-Chase detector.

$$\hat{a}_{k,l} = \begin{cases} s_l & , k = 1 \\ \mathrm{dec}\left\{ \left(z_k - \sum_{j<k} g_{k,j} \hat{a}_{j,l} \right) / g_{k,k} \right\} & , k = 2, \ldots, N \end{cases} \quad (5)$$

where dec{x} quantizes x to the nearest element of A, and $g_{k,j}$ is the element of $\mathbf{G}$ at the k-th row and j-th column. The DF detectors produce a set of candidate decision vectors $\{\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_q\}$, whose k-th elements are $\hat{a}_{k,l}$.

Step 4. The S-Chase detector chooses the candidate decision vector from step 3 with the minimum cost:

$$\arg\min_{l \in \{1, 2, \ldots, q\}} \| \boldsymbol{r} - \mathbf{H}\Pi \hat{\boldsymbol{a}}_l \|^2 = \arg\min_{l \in \{1, 2, \ldots, q\}} \| \boldsymbol{z} - \mathbf{G} \hat{\boldsymbol{a}}_l \|^2. \quad (6)$$

The latter is easier to implement because the l-th DF detector has already calculated most of the elements of $\mathbf{G}\hat{\boldsymbol{a}}_l$ in (5). Actually, the k-th element of $\boldsymbol{z} - \mathbf{G}\hat{\boldsymbol{a}}_l$ is simply a normalized version of the difference between the input and output of the k-th slicer.

Two simple enhancements reduce complexity even further. First, with the first candidate decision vector we can establish a *threshold* for the cost function. Subsequent DF detectors can be aborted whenever their cost exceeds the threshold. Second, the threshold can be reduced each time a candidate's cost is below the threshold. This shrinking threshold is reminiscent of a sphere detector [5], and suggests that we can reduce complexity by sorting the candidates according to their likelihood of being correct, i.e., according to their distance from $z_1/g_{1,1}$.

The pseudocode of an efficient implementation of the S-Chase detector is given in Fig. 3. Note that the lines (B-7), (B-12), and (B-13) may be omitted when $q = 1$, since in this case the one and only candidate vector is the final output.

Function SortedQR. Inputs: ($\mathbf{H}$, m); Outputs: ($\mathbf{Q}$, $\mathbf{G}$, $\boldsymbol{d}$, Π)

		Computations
(A-1)	$\Pi = \mathbf{I}_{N \times N}$	
(A-2)	$\mathbf{Q} = \mathbf{H}$	
(A-3)	$E_j = \sum_{k=1 \ldots M} \lvert q_{k,j} \rvert^2$, $j = 1$ to N,	Mag: MN, Add: $N(M-1)$
(A-4)	$i = \arg\max\{E_j^m : j = 1 \ldots N\}$	
(A-5)	$t = \max\{E_1, \ldots E_N\}$	
(A-6)	$E_i = 0$	
(A-7)	for $k = N$ downto 1	
(A-8)	if $k > 1$	
(A-9)	$i = \arg\min\{E_j : j = 1, \ldots k, E_j > 0\}$	
(A-10)	Swap the i-th and k-th columns of $\mathbf{Q}$, $\mathbf{G}$, and Π.	
(A-11)	Swap the i-th and k-th elements of E.	
(A-12)	else	
(A-13)	$E_k = E_k + t$	Add: 1
(A-14)	end	
(A-15)	$g_{k,k} = \sqrt{E_k}$	Sqrt: 1
(A-16)	$d_k = 1/g_{k,k}$	Div: 1
(A-17)	$\boldsymbol{q}_k = \boldsymbol{q}_k d_k$	$\mathrm{Mult_R}$: $2M$
(A-18)	for $j = k - 1$ downto 1	
(A-19)	$g_{k,j} = \boldsymbol{q}_k^* \boldsymbol{q}_j$	Mult: M, Add: $2M - 2$
(A-20)	$\boldsymbol{q}_j = \boldsymbol{q}_j - g_{k,j}\boldsymbol{q}_N$	Mult: M, Add: $2M$
(A-21)	$E_j = E_j - \lvert g_{k,j} \rvert^2$	Mag: 1, Add: 1
(A-22)	end	
(A-23)	end	

Fig. 2. A modified version of the sorted-QR decomposition [9]. The modifications are lines (A-4)-(A-6), and (A-13).

Function DF. Inputs: ($\boldsymbol{z}$, Π, q, $\mathbf{G}$, $\boldsymbol{d}$); Output: $\hat{\boldsymbol{a}}$

		Computations
(B-1)	$\boldsymbol{s} = [\, s_1 \ldots s_q \,]$, q symbols in A nearest $z_1 d_1$, sorted by increasing distance,	$\mathrm{Mult_R}$: 2, Add: q
(B-2)	$T = \infty$	
(B-3)	for $k = 1$ to q	
(B-4)	$f_{j,k} = g_{j,1} s_k$, $j = 1$ to N	$\mathrm{Mult_R}$: 2, Mult: $N-1$
(B-5)	$\hat{a}_{1,k} = s_k$	
(B-6)	$r_{j,k} = z_j - f_{j,k}$, $j = 1$ to N	Add: 2
(B-7)	$e_k = \lvert r_{1,k} \rvert^2$	Mag: 1
(B-8)	for $n = 2$ to N	
(B-9)	if $e_k < T$	
(B-10)	$t = \sum_{j=2}^{n-1} g_{n,j} \hat{a}_{j,k}$	Mult: $n-2$, Add: $2(n-3)$
(B-11)	$\hat{a}_{n,k} = \mathrm{dec}\{(r_{n,k} - t)\, d_n\}$,	$\mathrm{Mult_R}$: 2, Add: 2
(B-12)	$f_{n,k} = f_{n,k} + t + g_{n,n} \hat{a}_{n,k}$,	$\mathrm{Mult_R}$: 2, Add: 4
(B-13)	$e_k = e_k + \lvert z_n - f_{n,k} \rvert^2$	Mag: 1, Add: 3
(B-14)	end	
(B-15)	end	
(B-16)	if $e_k < T$	
(B-17)	$T = e_k$	
(B-18)	$\hat{a}_j = \hat{a}_{j,k}$, $j = 1$ to N	
(B-19)	end	
(B-20)	end	
(B-21)	$\hat{\boldsymbol{a}} = \Pi \hat{\boldsymbol{a}}$	

Fig. 3. Decision-feedback algorithm for the S-Chase detector.

A. The MMSE S-Chase Detector

The MMSE version of the S-Chase detector follows from typical MMSE derivations, which we omit here. We only show how the algorithms provided for the zero-forcing S-Chase detector can be used to implement its MMSE counterpart. The only difference is in the QR decomposition; steps 1 through 4 do not change. Specifically, instead of decomposing the channel matrix itself as in (3), the MMSE version is based on a decomposition of the concatenation of the channel matrix and the $\sigma\mathbf{I}$ matrix [10]:

$$\begin{bmatrix} \mathbf{H} \\ \sigma\mathbf{I} \end{bmatrix} \Pi = \begin{bmatrix} \mathbf{Q} \\ \mathbf{Q}_2 \end{bmatrix} \mathbf{G}, \tag{7}$$

where $\mathbf{Q}$ is an $M \times N$ matrix, and $\mathbf{G}$ is an $N \times N$ lower triangular matrix. The *SortedQR* function in Fig. 2 is easily modified to calculate $\mathbf{Q}$ and $\mathbf{G}$ from (7) instead of from (3); it requires only two changes. The first change is to input $[\mathbf{H}^T \sigma\mathbf{I}]^T$ instead of $\mathbf{H}$. With this input, the *SortedQR* function outputs the matrices $[\mathbf{Q}^T \mathbf{Q}_2{}^T]^T$ and $\mathbf{G}$. Therefore, the second change is to define the matrix $\mathbf{Q}$ in (7) as *the first M rows* of the first matrix output by the *SortedQR* function.

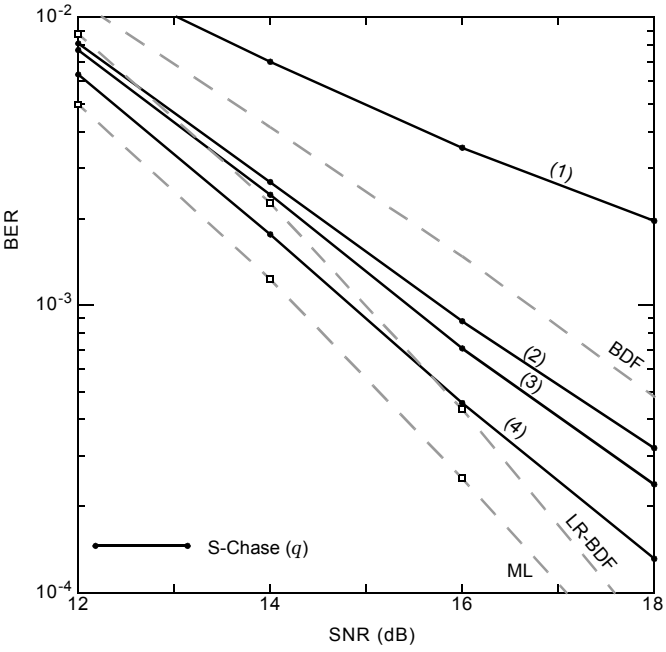

Fig. 4. Bit-error rate versus SNR for MMSE S-Chase(q) with $q \in \{1, 2, 3, 4\}$. The channel is 4×4 with 4-QAM inputs.

III. Numerical Results

In this section we quantify the performance and complexity of the proposed detector over a 4-input 4-output Rayleigh-fading channel. We assume $\mathbf{H}$ and σ are known perfectly by the receiver. We only present numerical results for the MMSE detectors, since they can significantly outperform their zero-forcing counterparts with little additional complexity. The SNR is defined as the average energy per signaling interval divided by the two-sided noise power spectral density at each receive antenna, $SNR = E[\|\mathbf{H}\boldsymbol{a}\|^2]/E[\|\boldsymbol{n}\|^2]$. All results are based on an average of 10^5 independent channel realizations. Throughout this section we use the notation S-Chase(q) to denote an S-Chase detector with parameter q, where q is both the length of the list and the number of DF detectors implemented in parallel.

We begin by showing how the performance of S-Chase(q) improves as q increases. Fig. 4 shows the bit-error rate (BER) of the S-Chase detector when the inputs are 4-QAM. Even small increases in q can lead to significant performance gains. For example, S-Chase(2) outperforms S-Chase(1) by 4.7 dB at a BER of 10^{-3}. A longer list length yields even better performance; S-Chase(4) outperforms S-Chase(1) by 5.6 dB.

Also shown in Fig. 4, for the sake of comparison, is the BER performance of the ML, MMSE BDF, and MMSE LR-BDF [6] detectors. We see that the S-Chase detector outperforms the MMSE-BDF detector whenever the list length is two or more, $q \geq 2$. The S-Chase(4) detector outperforms the LR-BDF detector of [6] at low SNR, when SNR < 16 dB, but for higher SNR the LR-BDF performs better.

The complexity of detection can be decomposed into preprocessing complexity and core-processing complexity. Preprocessing refers to those operations that are performed only once per channel estimation, while the core-processing is repeated every symbol period. We assume that the detector updates its channel estimate every L symbol periods, such that the core processing occurs L times for each occurrence of the preprocessing.

In order to quantify the performance-complexity trade-off, we define our measure of complexity as the number of real multiplications required in the worst case. Only real multiplications are counted to avoid defining the relative complexity of the different operations. This is a reasonable simplification since the total complexity is dominated by the number of multiplies. Each complex multiply is counted as three real multiplies, and each magnitude is counted as two real multiplies.

TABLE 1: WORST-CASE COMPLEXITY OF S-CHASE DETECTION WHEN $q > 1$.

	Number of Real Operations
ZF S-Chase	$3MN^2 + MN + N^2 - N + (3MN + 2)L + q(3N^2/2 + 9N/2 - 2)L$
MMSE S-Chase	$3MN^2 + N^3 + MN + N^2 - 2N + (3MN + 2)L + q(3N^2/2 + 9N/2 - 2)L$

It is easy to express the total number of real multiplications as a function of N, q, and L. Table 1 summarizes the total number of real multiplications required by the S-Chase detector assuming that all $q > 1$ costs must be completely calculated, i.e. $e_k < T$ is always true in (B-9). These complexity expressions follow in a straightforward manner from the algorithms given in Fig. 2 and Fig. 3, which also include the number of operations required at each line. To calculate $\mathbf{Q}$ and $\mathbf{G}$ (Fig. 2), the ZF S-Chase detector requires $3MN^2 + MN + N^2 - N$ multiplications. Calculating $\boldsymbol{z}$ in (4) requires $3MNL$ real multiplications. Finally, the DF detectors and the process of choosing the final decision (Fig. 3) require $q(3N^2/2 + 9N/2)L + 2L$ multiplications. Compared to its ZF counterpart, the only difference in implementing the MMSE S-Chase detector is in the calculation of $\mathbf{Q}$ and $\mathbf{G}$. For the MMSE implementation, the vector $\boldsymbol{q}_k$ in lines (A-17)-(A-20) has $M + N - k + 1$ non-zero elements so the number of real multiplies to find $\mathbf{Q}$ and $\mathbf{G}$ increases to $3MN^2 + N^3 + MN + N^2 - 2N$.

It is important to consider the value of L when comparing the complexity of different detectors. For example, the B-Chase detector [7] uses a better selection algorithm to achieve better performance than the S-Chase detector, but this requires more complex preprocessing. Therefore, the S-Chase detector can reduce overall complexity due to its low preprocessing complexity, but for large values of L it is more complex than B-Chase detection. Fig. 5 illustrates the relative complexity of the S-Chase and B-Chase detectors when the inputs are 4-QAM versus the frame length L. The main point of this graph is to demonstrate that when $q > 1$, the S-Chase detector is less complex than the B-Chase detector only for small L. Specifically, the S-Chase(2) detector is less complex than the BDF detector as long as $L < 5$. Also, the S-Chase(4) detector is less complex than the B-Chase(4) detector as long as $L < 9$.

A good means for comparing MIMO detectors is through their performance-complexity trade-off. Fig. 6 shows this trade-off for the S-Chase, B-Chase, and LR-BDF detectors for $L = 4$ when the inputs are 4-QAM. The performance measurement is the SNR these detectors need to reach BER 10^{-3}. For this frame length, the S-Chase(4) detector is 76% less complex than the LR-BDF detector while outperforming it by 0.2 dB. On the other hand, the B-Chase(4) detector outperforms the S-Chase(4) detector by 0.4 dB, but also increases complexity by 13%. The S-Chase detector has a particularly good performance-complexity trade-off when q is small, or when complexity is at a premium. For example, with $q = 2$ it outperforms the MMSE BDF detector by 1.1 dB while simultaneously reducing complexity by 6%. Not shown in Fig. 6 is the sphere detector, which achieves ML performance and has an SNR requirement of 14.1 dB, but is off the scale in complexity; it is six times as complex as S-Chase(4). Also not shown is the MMSE sphere detector, a near-ML detector with an SNR requirement of 14.1 dB that is more than three times as complex as S-Chase(4).

The performance-complexity trade-off of the S-Chase detector is a strong function of the alphabet size. Fig. 7 illustrates the trade-off with 16-QAM inputs. In this case, when $q > 4$, the B-Chase detector achieves better performance with less complexity than the S-Chase detector. However, the S-Chase detector maintains its advantage when q is small. For example, the S-Chase(2) detector outperforms the MMSE-BDF detector by 1.5 dB while simultaneously reducing complexity by 6%. Perhaps more impressive is S-Chase(4), which outperforms the MMSE-BDF detector by 5.1 dB while increasing complexity by only 30%. Not shown in Fig. 7 is the sphere detector, with an SNR requirement of 22 dB, which is 71 times as complex as the S-Chase(4) detector. Also not shown is the MMSE sphere detector, which requires an SNR of 22.5 dB, and which is 6.5 times as complex as the S-Chase(4) detector.

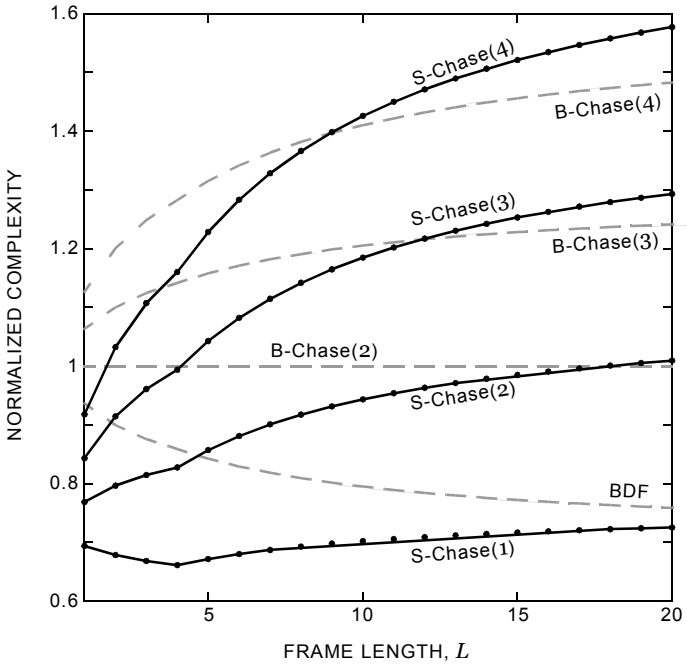

Fig. 5. Worst-case complexity of the MMSE S-Chase and B-Chase detectors over 4×4 channels with 4-QAM inputs. The complexity is measured as the number of multiplications, normalized by the complexity of B-

IV. Conclusions

We have proposed a new detection strategy for fading MIMO channels called the S-Chase detector, which combines a list detector and a parallel bank of suboptimally ordered DF detectors to achieve an attractive performance-complexity trade-off. For example, on a 4-input 4-output Rayleigh-fading channel with 16-QAM inputs, one version of the MMSE S-Chase detector outperforms the MMSE BLAST-ordered decision-feedback detector by 1.5 dB, while requiring 6% fewer computations.

V. References

[1] A. Chan, and I. Lee, "A New Reduced-Complexity Sphere Decoder For Multiple Antenna Systems," *IEEE Conf. Commun.*, vol. 1, pp. 460-464, May 2002.

[2] G. J. Foschini, G. Golden, R. Valenzuela, P. Wolniansky, "Simplified Processing for Wireless Communication at High Spectral Efficiency," *IEEE J. Select. Areas Commun.*, vol. 17, no. 11, pp. 1841-1852, 1999.

[3] W. J. Choi, R. Negi, and J. M. Cioffi, "Combined ML and DFE Decoding for the V-BLAST System," *IEEE Int. Conf. on Commun.*, vol. 3, pp. 1243-1248, June 2000.

[4] Y. Li, and Z. Luo, "Parallel Detection for V-BLAST System", *IEEE Conf. on Commun.*, vol. 1, pp. 340-344, May 2002.

[5] M. O. Damen, H. E. Gamal, and G. Caire, "On Maximum-Likelihood Detection and the Search for the Closest Lattice Point," *IEEE Trans. on Info. Th.*, vol. 49, no. 10, Oct. 2003.

[6] D. Wübben, R. Böhnke, V. Kühn, and K. Kammeyer, "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction," *IEEE Conf. on Commun.*, vol. 2, pp. 798-802, June. 2004.

[7] D. W. Waters and J. R. Barry, "The Chase Family of Detection Algorithms for Multiple-Input Multiple-Output Channels," *IEEE Global Commun. Conf.,* Nov. 2004.

[8] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Info. Th.*, vol. IT-18, pp. 170-182, Jan. 1972.

[9] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, "Efficient Algorithm for Decoding Layered Space-Time Codes," *Electronic Letters*, vol. 37, no. 22, pp. 1348-1350, Oct. 25, 2001.

[10] D. Wübben, R. Böhnke, V. Kühn, and K. Kammeyer, "MMSE Extension of V-BLAST based on Sorted QR Decomposition," *IEEE Veh. Tech. Conf.*, vol. 1, pp. 508-512, Oct. 2003.

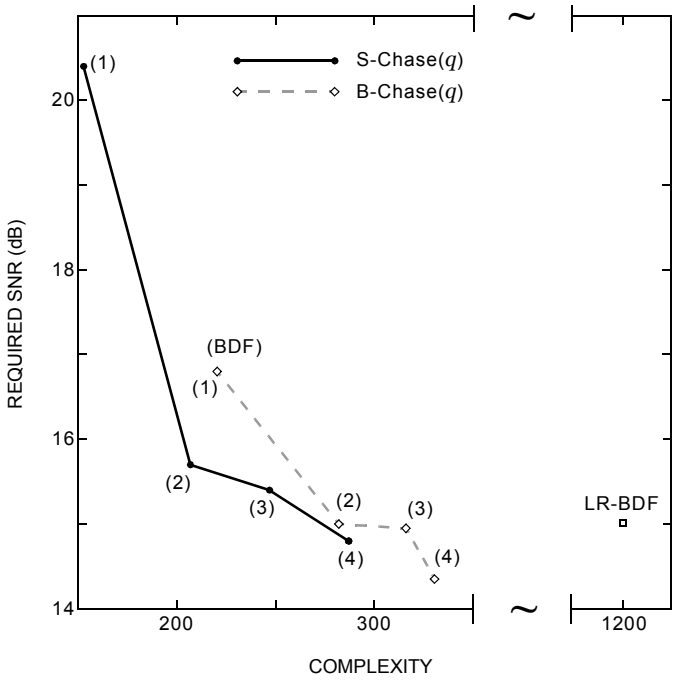

Fig. 6. Performance versus complexity for MMSE detectors with 4-QAM inputs. The results are averaged over 10^5 Rayleigh-fading 4×4 channels. The frame length is $L = 4$.

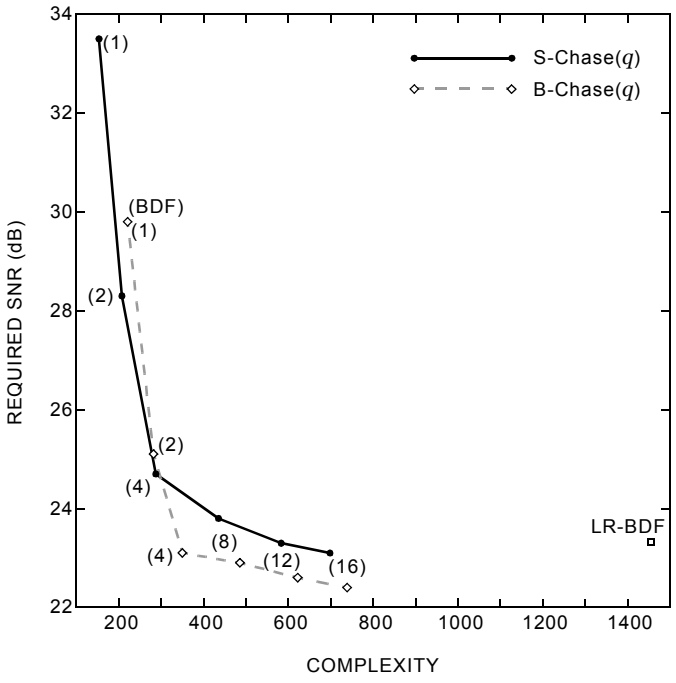

Fig. 7. Performance versus complexity for MMSE detectors with 16-QAM inputs. The results are averaged over 10^5 Rayleigh-fading 4×4 channels. The frame length is $L = 4$.