The Chase Family of Detection Algorithms for Multiple-Input Multiple-Output Channels

Deric W. Waters, *Member, IEEE*, and John R. Barry, *Senior Member, IEEE*

***Abstract*—We introduce a new family of detectors for multiple-input multiple-output (MIMO) channels. These detectors are called Chase detectors because they can be interpreted as a translation of the Chase error-control decoding algorithm from time to space. The Chase detector is parameterized by only four parameters; nevertheless, it reduces to a wide range of previously reported MIMO detectors as special cases, including the maximum-likelihood and decision-feedback detectors. The Chase detector defines a simple framework for not only comparing existing MIMO detection algorithms but also proposing new ones. For example, based on the Chase framework, we propose a new detector called B-Chase that performs well on fading channels. Specifically, on a four-input four-output Rayleigh-fading channel with uncoded 16-QAM inputs, one instance of the B-Chase detector falls only 0.4 dB short of the performance of the maximum-likelihood sphere detector while reducing complexity by 68%. Another instance of the B-Chase detector outperforms the BLAST-ordered decision-feedback detector by 4.4 dB while increasing complexity by only 17%.**

***Index Terms*—Complexity reduction, multiple- input multiple-output (MIMO) systems, signal detection, tree searching.**

I. Introduction

THE promise of high spectral efficiency and diversity to fading has led to widespread interest in multiple-input multiple-output (MIMO) communications. A practical obstacle to the realization of a MIMO system is the complexity of detection. For example, the complexity of maximum-likelihood (ML) detection grows exponentially with both the spectral efficiency and the number N of channel inputs. A popular reduced-complexity alternative, despite its significantly inferior performance, is the BLAST-ordered decision-feedback (BODF) detector [1]–[3], whose complexity is roughly independent of spectral efficiency and grows only cubically in N.

The large gap in both performance and complexity between the ML and BODF detectors has motivated the search for alternatives. The sphere detector [4] is a computationally efficient implementation of the ML detector. There has been extensive work to reduce its complexity [5], including the use of ordering [6]–[8] and optimization of the search radius [9]. A minimum mean-squared-error (MMSE) sphere detector was proposed in [6] that approximates the ML detector with reduced average complexity. Sphere detectors applied to a complex channel model have also been proposed [10], [11]. Other reduced-complexity approximations of the ML detector have also been proposed [12]–[18]. Lattice reduction [19], [20] also improves the performance of the BODF detector. A combination of Lenstra–Lenstra–Lovász (LLL) lattice reduction with the MMSE BODF detector closely approximates the ML detector in some cases [21].

There is an important class of reduced-complexity detectors called *list-based detectors* that adopts a two-step approach of first creating a list of candidate decision vectors, then choosing the best candidate as its final decision. The Chase detector introduced in this paper is an example of a list-based detector; other examples include [14], [15], [18], [22], and [23]. The rollout detector of [14] enumerates all possibilities for the first m symbols, then completes the decision vector for each possibility using a DF detector. The parallel detector [15] generates its list by implementing a separate low-complexity detector for each possible value of the first symbol. In fact, the parallel detector is like a rollout detector with $m = 1$, except that it uses a unique symbol ordering to improve performance. More recently, a generalization of the parallel detector called the fixed-complexity sphere detector has been shown to achieve full diversity over N-input N-output channels [23].

This paper proposes the *B-Chase detector*, and demonstrates that the B-Chase detector can approach ML performance in some cases with less complexity than previously reported detectors [6], [12], [15], [21]. The B-Chase detector distinguishes itself from previous list-based detectors in the unique way it builds its list. It will be shown that the B-Chase detector achieves better performance with significantly smaller list lengths, leading to a favorable performance-complexity tradeoff for four-input four-output channels. Instead of enumerating all possibilities for the first symbol, like the parallel and rollout detectors, the B-Chase detector may enumerate only a subset of the possible symbol values. Furthermore, the B-Chase detector adopts a unique symbol ordering, which is critical to its performance.

In Section II, we introduce the Chase framework for defining detection algorithms, and show how existing detectors fit into the framework. In Section III, we propose a new instance of the Chase detector family called the B-Chase detector. In

Manuscript received May 23, 2006; revised July 30, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sergiy Vorobyov. This research was supported in part by National Science Foundation grants 0431031 and 0121565. Portions of this work were presented in the *Proceedings of the IEEE Global Telecommunications Conference* (IEEE GLOBECOM), Dallas, TX, November 29–December 3, 2004, vol. 4, pp. 2635–2639.

D. W. Waters is with Texas Instruments, Dallas, TX 75243 USA (e-mail: deric@ti.com).

J. R. Barry is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: barry@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2007.907904

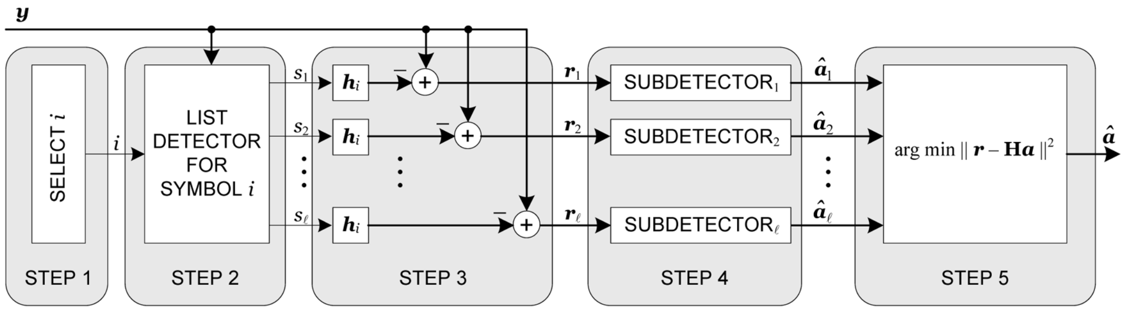


Fig. 1. Block diagram of the Chase detector.

Section IV, we describe a computationally efficient implementation of the B-Chase detector. In Section V, we present some performance and complexity numerical results, and in Section VI, we make concluding remarks.

II. Chase Detection: A General Framework

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

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

where $\mathbf{H} = [\boldsymbol{h}_1, \dots \boldsymbol{h}_N]$ is a complex $M \times N$ channel matrix whose ith column is $\boldsymbol{h}_i$, and where $\boldsymbol{w} = [w_1, \dots w_M]^T$ is noise. We assume that the columns of $\mathbf{H}$ are linearly independent, which implies $M \geq N$. We assume that the noise components are independent and identically distributed (i.i.d.) complex Gaussian random variables with $\mathrm{E}[\boldsymbol{w}\boldsymbol{w}^*] = N_0\mathbf{I}$, where $\boldsymbol{w}^*$ denotes the conjugate transpose of $\boldsymbol{w}$. Further, we assume that the complex inputs are uncorrelated and chosen from the same unit-energy discrete alphabet A, so that $\mathrm{E}[\boldsymbol{a}\boldsymbol{a}^*] = \mathbf{I}$.

In this section, we introduce the Chase detector, a general detection strategy for MIMO channels that reduces to a variety of previously reported detectors as special cases. The Chase detector defines a simple framework for not only comparing existing MIMO detection algorithms but also proposing new ones. Specifically, a Chase detector is defined by five steps, as illustrated in Fig. 1, and as outlined below.

Step 1) Identify $i \in \{1, \dots N\}$, the index of the first symbol to be detected.

Step 2) Generate a sorted list $\mathcal{L}$ of candidate values for the ith symbol, defined as the ℓ elements of the alphabet nearest to y_i, where $\boldsymbol{y} = (\mathbf{H}^*\mathbf{H} + \alpha^2\mathbf{I})^{-1}\mathbf{H}^*\boldsymbol{r}$ is the output of either the zero-forcing (ZF) $(\alpha = 0)$ or MMSE $(\alpha^2 = N_0)$ linear filter.

Step 3) Generate a set of ℓ residual vectors $\{\boldsymbol{r}_1, \dots \boldsymbol{r}_\ell\}$ by cancelling the contribution to $\boldsymbol{r}$ from the ith symbol, assuming each candidate from the list is, in turn, correct:

$$\boldsymbol{r}_j = \boldsymbol{r} - \boldsymbol{h}_i s_j. \tag{2}$$

Step 4) Apply each of $\{\boldsymbol{r}_1, \dots \boldsymbol{r}_\ell\}$ to its own independent subdetector, which makes decisions about the remaining $N - 1$ symbols (all but the ith symbol). Together with s_j, the jth subdetector defines a *candidate* hard decision $\hat{\boldsymbol{a}}_j$ regarding the input $\boldsymbol{a}$.

Step 5) Choose as the final hard decision $\hat{\boldsymbol{a}}$ the candidate hard decision $\{\hat{\boldsymbol{a}}_1, \dots \hat{\boldsymbol{a}}_\ell\}$ that best represents the observation $\boldsymbol{r}$ in a minimum mean-squared-error sense:

$$\hat{\boldsymbol{a}} = \underset{\boldsymbol{a} \in \{\hat{\boldsymbol{a}}_1, \dots \hat{\boldsymbol{a}}_\ell\}}{\arg\min} \|\boldsymbol{r} - \mathbf{H}\boldsymbol{a}\|^2. \tag{3}$$

The Chase detector is roughly analogous to its namesake, the well-known Chase algorithm for soft decoding of binary error-control codes [24], but with the temporal dimension replaced by the spatial dimension. The analogy is loose, but still useful. The Chase algorithm begins by identifying the p least reliable bits of a received codeword, and enumerates all 2^p corresponding binary vectors while fixing the remaining more reliable bits. This is analogous to Steps 1) and 2), except in Step 1), only *one* symbol (not necessarily the least reliable) is identified instead of p, and in Step 2), only a subset of the most likely values are enumerated. The Chase algorithm decodes each of the 2^p binary vectors using a simple hard-decoding algorithm, producing a set of candidate hard decisions for the codeword. This is analogous to the cancellation and subdetection in Steps 3) and 4). Finally, the Chase algorithm chooses the candidate codeword that best matches the received observations in a way precisely analogous to that in Step 5).

To uniquely define an instance of the Chase detector requires that the following four parameters be specified:

- a strategy for selecting i in Step 1);
- a list length ℓ for Step 2);
- a filter type, ZF or MMSE, for Step 2);
- a subdetector algorithm for Step 4).

Table I summarizes how the maximum-likelihood (ML), BODF, parallel decision feedback (PDF), and parallel detectors may be specified as Chase detectors using these four parameters. For example, the Chase detector reduces to the ML detector when the subdetectors are themselves ML detectors, and the list length is maximal ($\ell = |A|$). In this case, the choice of which symbol to detect first has no effect on performance. On the other hand, the Chase detector reduces to the BODF detector when the list length is one and the subdetectors are themselves BODF detectors. In this case, the choice of which symbol to detect first is critical to performance. The parallel detector is another Chase detector whose performance is highly sensitive to the choice of which symbol to detect first. The last row of Table I describes a new detector that will be proposed in the next section.

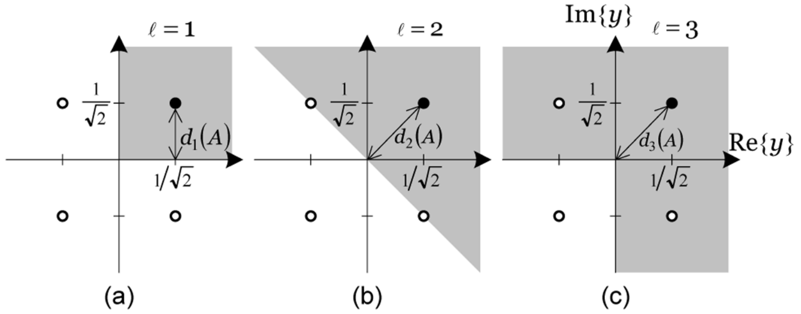


Fig. 2. Decision regions for $a = e^{j\pi/4}$ and different list lengths: (a) $\ell = 1$; (b) $\ell = 2$; and (c) $\ell = 3$. The decision list contains a whenever the input to the list detector falls within the shaded region. Also indicated is the minimum distance d_ℓ to the boundary.

TABLE I
SPECIAL CASES OF THE CHASE DETECTOR

Detector	First-Symbol Index i	List Length ℓ	Filter Type, α	Subdetector
ML [5]	any	$\lvert A\rvert$	ZF	ML
BODF [1]	[a] BLAST$_1$	1	ZF or MMSE	BODF
PDF [23]	[a] BLAST$_1$	1	ZF or MMSE	Linear
Parallel [15]	using (14)	$\lvert A\rvert$	ZF	any
B-CHASE	using (14) or (16)	$1 \le \ell \le \lvert A\rvert$	ZF or MMSE	BODF

[a] The index BLAST_1 signifies the first index of the BLAST ordering [1].

III. A NEW CHASE DETECTOR

In this section, we introduce the B-Chase detector, as summarized by the last row of Table I. The B-Chase detector is defined simply as a Chase detector that uses BODF as a subdetector. The list length ℓ can be any integer in the set $\{1, \ldots |A|\}$, and the filters can be ZF or MMSE. It remains to specify the key parameter, namely, the index i of the symbol to detect first. Two algorithms for selecting i will be described later in this section. Before describing them, we must first understand the impact of the list detector on the signal-to-noise ratio (SNR) of the ith symbol.

A. SNR Gain of a List Detector

We say that a list detector makes an *error* when the actual transmitted symbol does not appear somewhere on the list. With this definition, increasing the length of the list leads to a decrease in the probability of error. (Indeed, a maximal list length of $|A|$ ensures that the list detector *never* makes an error.) The decrease in error probability can be interpreted as an SNR gain. We demonstrate this effective gain using the 4-QAM alphabet as an example.

Assume a 4-QAM alphabet $\{e^{\pm j\pi/4}, e^{\pm j3\pi/4}\}$ with a ZF front end $(\alpha = 0)$, and assume that the transmitted symbol is $a = e^{j\pi/4}$. The input to the list detector is then $y_i = a + n$; this defines a scalar channel whose SNR is $\mathrm{SNR}_i = 1/E[|n|^2]$. In Fig. 2, we illustrate the correct decision regions for lists lengths $\ell \in \{1, 2, 3\}$. As shown in Fig. 2(a), a list detector with $\ell = 1$ (i.e., a conventional decision device) will be correct when y_i is in the first quadrant of the complex plane. As illustrated in Fig. 2(b), a list detector with $\ell = 2$ will be correct when $\mathrm{Re}\{y_i\} + \mathrm{Im}\{y_i\} > 0$. As illustrated in Fig. 2(c), a list detector with $\ell = 3$ will be correct when y_i is not in the third quadrant. Therefore, letting P_ℓ denote the list-error probability for 4-QAM when the list length is ℓ, we find that

$$\begin{aligned} P_1 &= 2Q(\sqrt{\mathrm{SNR}_i}) - Q^2(\sqrt{\mathrm{SNR}_i}) \\ &\approx 2e^{-\mathrm{SNR}_i/2} - e^{-\mathrm{SNR}_i} \\ &\approx e^{-\mathrm{SNR}_i/2}, \qquad (4) \\ P_2 &= Q(\sqrt{2\mathrm{SNR}_i}) \approx e^{-\mathrm{SNR}_i} \qquad (5) \\ P_3 &= Q^2(\sqrt{\mathrm{SNR}_i}) \approx e^{-\mathrm{SNR}_i} \qquad (6) \end{aligned}$$

where $Q(x) = (2\pi)^{-1/2} \int_x^\infty e^{-t^2/2} dt$. We twice invoked the Chernoff-bound approximation $Q(x) \approx e^{-x^2/2}$, which is valid only at high SNR, and we further assumed that $\mathrm{SNR}_i \gg \log(4)$ in the second approximation for P_1. Comparing (4) and (5), we see that increasing the list length from $\ell = 1$ to $\ell = 2$ approximately doubles the SNR. Intuitively, we can attribute this SNR gain to the fact that the minimum distance to the decision boundary increases by a factor of $\sqrt{2}$ when the list length is increased from $\ell = 1$ to $\ell = 2$. Likewise, the SNR gain for $\ell = 3$ is the same as that for $\ell = 2$ because, as shown in Fig. 2, they have the same minimum distance from a to the decision boundary.

We approximate the list detector SNR gain at high SNR by how far it moves the decision boundary. Specifically, let $d_\ell(A)$ denote the minimum distance from any element in A to the corresponding decision region boundary of the list detector with list length ℓ. We define the SNR *gain* γ_ℓ^2 for a list detector with a list length of ℓ as

$$\gamma_\ell^2 = \frac{d_\ell^2(A)}{d_1^2(A)}. \qquad (7)$$

This gain approximately quantifies the benefit of a list detector $(\ell \ge 1)$ relative to a conventional detector $(\ell = 1)$. For example, from Fig. 2 we see that d_1(4-QAM) $= 1/\sqrt{2}$ and d_2(4-QAM) $= d_3$(4-QAM) $= 1$, which is consistent with an SNR gain of two for both $\ell = 2$ and $\ell = 3$. At one extreme, a minimal list length $(\ell = 1)$ yields no SNR gain $(\gamma_1^2 = 1)$, as expected. At the other extreme, a maximal list length $(\ell = |A|)$ yields an infinite SNR gain $(\gamma_{|A|}^2 = \infty)$, since there is no decision boundary at all in that case. A straightforward analysis of the list detector decision regions for 16-QAM reveals that the list detector SNR gains are $\gamma_2^2 = 2$, $\gamma_3^2 = 2$, $\gamma_8^2 = 8$, and $\gamma_{10}^2 = 10$. Similarly, the list detector SNR gains for 64-QAM are $\gamma_4^2 = 4$, $\gamma_8^2 = 8$, $\gamma_{18}^2 = 20$, $\gamma_{33}^2 = 40$, and $\gamma_{48}^2 = 58$. When compared to the true SNR gain of a list detector, as measured at an error probability of 0.01, the approximation of (7) is accurate to within 1 dB for a 16-QAM list detector with list length $\ell \in \{1,\dots 9\}$, and it is accurate to within 1 dB for a 64-QAM list detector with list length $\ell \in \{1,\dots 41\}$.

B. SNR of the B-Chase Detector

In this subsection we quantify the SNR for each symbol of the B-Chase detector. We begin by analyzing the output of the linear filter in Step 2) of the B-Chase detector, which provides the input to the list detector. First, consider the QR decomposition of the extended channel matrix [25], [26]:

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \alpha \mathbf{I} \end{bmatrix} = \widetilde{\mathbf{Q}}\mathbf{L} \tag{8}$$

where the columns of the $(M+N) \times N$ matrix $\widetilde{\mathbf{Q}}$ are orthonormal, and where $\mathbf{L}$ is a lower triangular $N \times N$ matrix with positive and real diagonal elements. The bottom N rows of $\widetilde{\mathbf{Q}}$ are the matrix $\alpha \mathbf{L}^{-1}$ [26].

In terms of the QR decomposition (8), the linear filter of Step 2) can be written as $\mathbf{U}^*\mathbf{Q}^*$, where $\mathbf{U}^* = \mathbf{L}^{-1}$, and where the matrix $\mathbf{Q}$ is defined as the top M rows of $\widetilde{\mathbf{Q}}$. The output of this linear filter is thus $\boldsymbol{y} = \mathbf{U}^*\mathbf{Q}^*\boldsymbol{r}$, which reduces to

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{a} - \alpha^2 \mathbf{U}^*\mathbf{U}\boldsymbol{a} + \mathbf{U}^*\mathbf{Q}^*\boldsymbol{w} \\ &= \boldsymbol{a} + \boldsymbol{n} \end{aligned} \tag{9}$$

where we used the fact that $\mathbf{Q}^*\mathbf{H} = \mathbf{L} - \alpha^2\mathbf{U}$, and where we introduced $\boldsymbol{n} = \mathbf{U}^*\mathbf{Q}^*\boldsymbol{w} - \alpha^2\mathbf{U}^*\mathbf{U}\boldsymbol{a}$. Although $\boldsymbol{n}$ contains both noise (first term) and residual intersymbol interference (ISI) (second term) when $\alpha \neq 0$, we continue to call it *noise*. Since $\mathbf{Q}^*\mathbf{Q} + \alpha^2\mathbf{U}\mathbf{U}^* = \mathbf{I}$ and $\alpha^2 \in \{0, N_0\}$, the noise variance of the ith output of the forward filter is $\mathrm{E}[|n_i|^2] = N_0\|\boldsymbol{u}_i\|^2$, where $\boldsymbol{u}_i$ is the ith column of $\mathbf{U}$. There is a slight bias when $\alpha = N_0$ (MMSE), but to keep complexity low, we opt not to remove it.

The effective SNR for the first symbol detected, including the gain of the list detector, is

$$\mathrm{SNR}_1^{(i)} = \frac{\gamma_\ell^2}{N_0\|\boldsymbol{u}_i\|^2}. \tag{10}$$

A more convenient expression for $\mathrm{SNR}_1^{(i)}$, and for the SNRs of the remaining symbols, is defined by a QR decomposition of the extended channel matrix $\overline{\mathbf{H}}$ whose columns are permuted according to the detection order. Let $\Pi^{(i)}$ denote an $N \times N$ permutation matrix that arranges the columns of $\overline{\mathbf{H}}$ such that the ith column comes first, and the remaining columns are arranged according to the BLAST ordering. Consider the QR decomposition

$$\overline{\mathbf{H}}\Pi^{(i)} = \widetilde{\mathbf{Q}}^{(i)}\mathbf{L}^{(i)} \tag{11}$$

where the columns of the $(M+N) \times N$ matrix $\widetilde{\mathbf{Q}}^{(i)}$ are orthonormal, and where $\mathbf{L}^{(i)}$ is a lower triangular $N \times N$ matrix with positive and real diagonal elements. Note that $\widetilde{\mathbf{Q}} = \widetilde{\mathbf{Q}}^{(i)}$ and $\mathbf{L} = \mathbf{L}^{(i)}$ only when $\Pi^{(i)}$ is the identity matrix. The effective SNR for the first symbol detected is

$$\mathrm{SNR}_1^{(i)} = \frac{\gamma_\ell^2 \cdot \left(l_{1,1}^{(i)}\right)^2}{N_0} \tag{12}$$

where $l_{k,k}^{(i)}$ is the kth diagonal of $\mathbf{L}^{(i)}$. The final $N-1$ symbols in the B-Chase detector when the ith symbol is detected first do not enjoy any list-detection gain. Therefore, assuming no error propagation, their SNR can be expressed as

$$\mathrm{SNR}_k^{(i)} = \frac{\left(l_{k,k}^{(i)}\right)^2}{N_0}, \quad k = 2, \dots, N. \tag{13}$$

This assumption is justified by the fact that error propagation is not the limiting factor of performance, even for small list lengths.

C. B-Chase Selection

The importance of which symbol is detected first is greatly impacted by the list length. Consider two extreme cases: First, when the list length is maximal ($\ell = |A|$), the *least* reliable symbol should be detected first. (The Chase error-control decoding algorithm similarly identifies the least reliable bits to enumerate.) When the list length is minimal ($\ell = 1$), the *most* reliable symbol should be detected first. (This is consistent with BLAST-ordered DF detection.) In between these two extremes, however, the choice of which symbol to detect first must balance two opposing goals. On the one hand, we want to choose i so that the SNR of the first symbol $\mathrm{SNR}_1^{(i)}$ is high; this ensures that the list detector is likely to be correct. Loosely speaking, $\mathrm{SNR}_1^{(i)}$ is maximized by choosing the column of $\overline{\mathbf{H}}$ that is *most orthogonal* to the remaining columns. On the other hand, we also want each subdetector to see a well-conditioned channel, so that the subdetector decisions are likely to be correct. Loosely speaking, this is accomplished by choosing the column of $\overline{\mathbf{H}}$ that is *least* orthogonal to the remaining columns. We now describe two selection algorithms that strike a balance between these two opposing goals.

Selection Algorithm 1: Our first selection algorithm maximizes the minimum SNR of the symbols, as follows:

$$i = \underset{k \in \{1,2,\dots N\}}{\arg\max} \left\{ \gamma_\ell \cdot l_{1,1}^{(k)}, l_{2,2}^{(k)}, \dots, l_{N,N}^{(k)} \right\}. \tag{14}$$

When $\ell = 1$, so that $\gamma_\ell = 1$, this selection algorithm can be implemented by choosing the column of $\mathbf{U}$ with minimum norm, as proven in [1]. On the other hand, when the list length is maximal and $\gamma_\ell = \infty$, the selection algorithm reduces to the parallel selection algorithm [15].

Implementing the selection algorithm (14) when $\ell > 1$ requires $O(N^4)$ computations. This is because the QR decomposition (11) needs to be computed N times, where each decomposition involves computing the BLAST ordering of an $M \times (N-1)$ matrix.

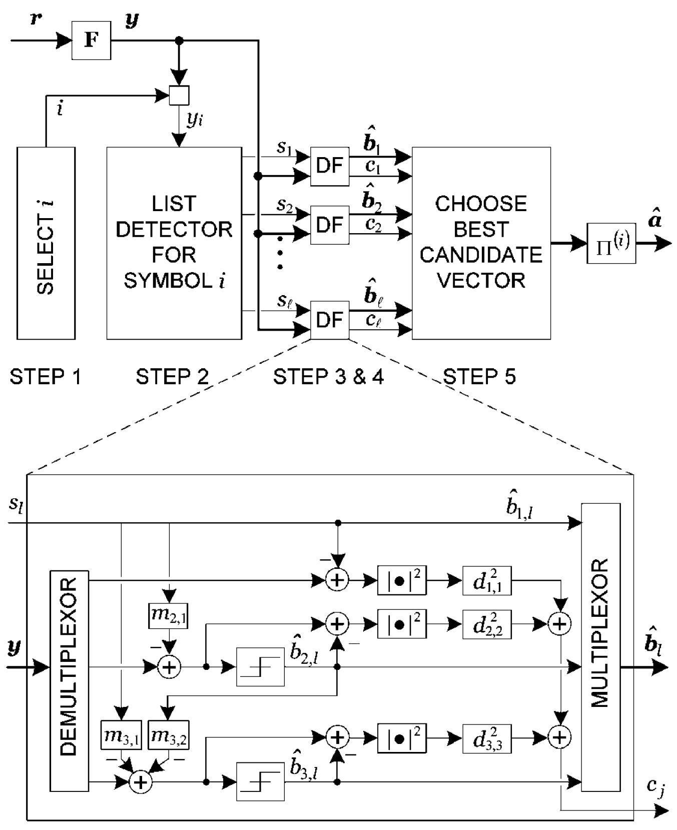


Fig. 3. (a) Overall block diagram for the B-Chase detector. (b) Block diagram for the DF subdetector when $N = 3$.

Selection Algorithm 2: In order to avoid the large complexity of the first selection algorithm, we propose approximating the SNR of the symbols inside the subdetectors. First of all, if $\ell = 1$ we select the symbol with minimum noise variance, because this is optimal [1]. On the other hand, if the list length is maximal ($\ell = |A|$), we select the symbol with the largest noise variance because the list detector has an infinite SNR gain to counteract the noise. When the list length is greater than one, but not maximal, we propose selecting the symbol which maximizes the minimum of $\text{SNR}_1^{(i)}$ and $\text{SNR}_2^{(i)}$. This approach is justified by the fact that the smallest SNR inside the subdetector is often $\text{SNR}_2^{(i)}$, so that $\text{SNR}_2^{(i)}$ approximates the minimum SNR inside the subdetector. The ultimate validity of this approximation will be shown through the performance results of Section V. This SNR can be easily calculated from the matrix $\mathbf{U}$, as follows:

$$\text{SNR}_2^{(i)} = \frac{1}{N_0 \min_{j \neq i} \{\|\boldsymbol{u}_j\|^2 - |g_{j,i}|^2\}} \tag{15}$$

where $g_{j,i} = \boldsymbol{u}_j^* \boldsymbol{u}_i / \|\boldsymbol{u}_i\|$. Selection algorithm 2 can thus be summarized as follows:

$$i = \begin{cases} \underset{k \in \{1,2,\ldots N\}}{\arg\max} \ \|\boldsymbol{u}_k\|^2, & \ell = |A| \\ \underset{k \in \{1,2,\ldots N\}}{\arg\max} \ \min\left\{ \frac{\gamma_\ell^2}{\|\boldsymbol{u}_k\|^2}, \frac{1}{\min_{j \neq k}\{\|\boldsymbol{u}_j\|^2 - |g_{j,k}|^2\}} \right\}, & \text{else.} \end{cases} \tag{16}$$

Note that if $\ell = 1$, (16) reduces to choosing the column of $\mathbf{U}$ with minimum norm.

FUNCTION BCHASE DETECTOR

INPUTS: $\overline{\mathbf{H}}, \ell, \mathcal{A}$
OUTPUT: $\hat{\boldsymbol{a}}$

```
1.  $[\mathbf{F}, \mathbf{M}, \Pi^{(i)}, \{d_{j,j}^2\}] = \text{BChasePreprocessing}(\overline{\mathbf{H}}, \ell)$
2.  $\boldsymbol{y} = \mathbf{F}\boldsymbol{r}$
3.  $[s_1, \ldots s_\ell] = \text{ListDetect}(y_1 | \mathcal{A}, \ell)$,
    so that $s_i$ is the $i$-th closest element of $\mathcal{A}$ to $y_1$
4.  $\tau = \infty$
5.  for $l = 1$ to $\ell$,
6.      $\hat{b}_l = s_l$
7.      $c_l = d_{1,1}^2 |y_1 - s_l|^2$
8.      for $k = 2$ to $N$,
9.          if $c_l < \tau$,
10.             $x = y_k - \sum_{j=1}^{k-1} m_{k,j} \hat{b}_{j,l}$
11.             $\hat{b}_{k,l} = \text{dec}\{x\}$
12.             $c_l = c_l + d_{k,k}^2 |x - \hat{b}_{k,l}|^2$
13.         end
14.     end
15.     if $c_l < \tau$,
16.         $\tau = c_l$
17.         $f = l$
18.     end
19. end
20. $\hat{\boldsymbol{a}} = \Pi^{(i)} \boldsymbol{b}_f$
```

Fig. 4. Computationally efficient implementation of the B-Chase detector.

IV. IMPLEMENTING THE B-CHASE DETECTOR

In this section, we describe an implementation of the B-Chase detector that has low complexity, as illustrated by the block diagram of Fig. 3, and as summarized by the pseudocode of Figs. 4 and 5. This computationally efficient implementation enables a detailed performance-complexity tradeoff analysis in Section V.

Step 1): The first step towards implementing the B-Chase detector is to select the symbol to detect first according to (14) or (16). Selection algorithm 1 can be implemented directly once the squares of the diagonal elements of $L^{(i)}$ from (11) are known. We will calculate these without computing the QR decomposition of (11) directly. Observe that permuting the *columns* of $\overline{\mathbf{H}}$ by $\Pi^{(i)}$ corresponds to permuting the *rows* of $\mathbf{C} = \mathbf{U}^* \widetilde{\mathbf{Q}}^*$ by $\Pi^{(i)*}$. As a result, the definitions of $\Pi^{(i)}$, $\widetilde{\mathbf{Q}}^{(i)}$, and $\mathbf{L}^{(i)}$ given in (11) are equivalently defined by the following sorted-QR decomposition of $\mathbf{C}^*$:

$$\mathbf{C}^* \Pi^{(i)} = \widetilde{\mathbf{Q}}^{(i)} \mathbf{U}^{(i)} \tag{17}$$

where $\mathbf{U}^{(i)} = (\mathbf{L}^{(i)*})^{-1}$. This sorted-QR decomposition can be computed using the algorithm given in [27] after modifying it to choose the ith column first, then choose the following columns

```
FUNCTION BCHASEPREPROCESSING
    INPUTS: $\overline{\mathbf{H}}$, $\ell$
    OUTPUTS: $\mathbf{F}$, $\mathbf{M}$, $\Pi^{(i)}$, $\{d_{1,1}^2, \ldots d_{N,N}^2\}$
1.  $[\tilde{\mathbf{Q}}, \mathbf{L}] = \mathrm{QRdecomposition}(\overline{\mathbf{H}})$
2.  $\mathbf{U} = (\mathbf{L}^*)^{-1}$
3.  for $j = 1$ to $N$,    $e_j = \sum_{k \in \{1, \ldots, j\}} |u_{k,j}|^2$,    end
4.  for $k = 1$ to $N$,
5.      $[\Theta^{(k)}, \mathbf{U}^{(k)}, \Pi^{(k)}, \{(l_{1,1}^{(k)})^2, \ldots, (l_{N,N}^{(k)})^2\}] = \mathrm{sortedQR}(\mathbf{U}, \boldsymbol{e}, k)$
6.      $S^{(k)} = \min\{(\gamma_\ell l_{1,1}^{(k)})^2, (l_{2,2}^{(k)})^2, \ldots, (l_{N,N}^{(k)})^2\}$
7.  end
8.  $i = \arg\max_{k \in \{1, \ldots, N\}} S^{(k)}$
9.  $\mathbf{D}^{-1} = \mathrm{diag}(\mathbf{U}^{(i)})$
10. $\mathbf{Q} =$ first $M$ rows of $\tilde{\mathbf{Q}}$
11. $\mathbf{F} = \mathbf{D}^{-1}\Theta^{(i)*}\mathbf{Q}^*$
12. $\mathbf{M} = \mathbf{D}^{-1}\Theta^{(i)*}\mathbf{L}\Pi^{(i)}$
13. for $j = 1$ to $N$,    $d_{j,j}^2 = (l_{j,j}^{(i)})^2$,    end
```

Fig. 5. Preprocessing pseudocode for the proposed implementation of the B-Chase detector that uses selection algorithm 1.

normally. It is important to note that $\Pi^{(i)}$ calculated in this way puts the final $N-1$ columns of $\overline{\mathbf{H}}$ in their BLAST ordering, as shown in [28]. Finally, the squares of the diagonal elements of $\mathbf{L}^{(i)}$ are a by-product of this sorted-QR decomposition, and selection algorithm 1 can be implemented using (14).

Lines 3–8 of Fig. 5 implement selection algorithm 1 in a less complex way by computing the sorted-QR decomposition of the lower triangular matrix $\mathbf{U}$, as we now explain. First, substituting the definition of $\mathbf{C}$ into (17) gives:

$$\begin{aligned}\mathbf{C}^*\Pi^{(i)} &= \widetilde{\mathbf{Q}}\mathbf{U}\Pi^{(i)} \\ &= \widetilde{\mathbf{Q}}\Theta^{(i)}\Theta^{(i)*}\mathbf{U}\Pi^{(i)}\end{aligned} \tag{18}$$

where $\Theta^{(i)}$ is a unitary matrix such that $\mathbf{U}^{(i)} = \Theta^{(i)*}\mathbf{U}\Pi^{(i)}$ is an upper triangular matrix with real and positive diagonals. Then by inspection we see that $\widetilde{\mathbf{Q}}^{(i)} = \widetilde{\mathbf{Q}}\Theta^{(i)}$. The matrices $\Theta^{(i)}$, $\mathbf{U}^{(i)}$, and $\Pi^{(i)}$ are simply defined by the sorted-QR decomposition of $\mathbf{U}$

$$\mathbf{U}\Pi^{(i)} = \Theta^{(i)}\mathbf{U}^{(i)}. \tag{19}$$

As before, the squares of the diagonal elements of $\mathbf{L}^{(i)}$ are a by-product of this decomposition.

Before moving on to Step 2), we propose applying a front-end filter to the channel output that reduces the complexity of subsequent steps. Lines 9–11 of Fig. 5 give the pseudocode for computing the front-end filter $\mathbf{F}$, which is defined as follows:

$$\mathbf{F} = \mathbf{D}^{-1}\mathbf{Q}^{(i)*} \tag{20}$$

where $\mathbf{D}$ is a diagonal matrix with $d_{j,j} = l_{j,j}^{(i)}$. Similar to (9), the output of this filter $\boldsymbol{y} = \mathbf{F}\boldsymbol{r}$ reduces to

$$\boldsymbol{y} = \mathbf{M}\boldsymbol{b} + \boldsymbol{n} \tag{21}$$

where $\mathbf{M} = \mathbf{D}^{-1}\mathbf{L}^{(i)}$ is an $N \times N$ lower-triangular matrix with ones along the diagonal, where $\boldsymbol{b} = \Pi^{(i)*}\boldsymbol{a}$ is a permuted version of the channel input, and the effective noise is $\boldsymbol{n} = \mathbf{F}\boldsymbol{w} - \alpha^2\boldsymbol{D}^{-1}\mathbf{U}^{(i)}\boldsymbol{b}$. Line 12 of Fig. 5 gives the pseudocode for computing $\mathbf{M}$, which will be needed to implement the subdetectors.

Step 2): After applying the front-end filter as shown in Line 2 of Fig. 4 to compute $\boldsymbol{y}$ (21), the list detector simply generates an ordered list $[s_1, \ldots s_\ell]$ of the ℓ elements of A that are nearest to y_1.

Steps 3) and 4): It is convenient for implementation to merge Steps 3) and 4). The result is ℓ DF detectors whose first symbol decisions are hard-wired to distinct outputs of the list detector. Using the well-known decision-feedback process [29], the lth subdetector cancels the intersymbol interference from the kth element of $\boldsymbol{y}$ as follows:

$$x_{k,l} = y_k - \sum_{j=1}^{k-1} m_{k,j}\hat{b}_{j,l} \tag{22}$$

where $\hat{b}_{j,l} = \mathrm{dec}\{x_{k,l}\}$ is the decision already made regarding b_j by the ℓth subdetector, and where $\mathrm{dec}\{z\}$ quantizes z to the nearest element of A.

Step 5): In the fifth and final step, the B-Chase detector chooses its final decision as the subdetector's output which has the minimum cost. From (3), the cost of the ℓth decision vector can be expressed as $c_l = \|\boldsymbol{r} - \mathbf{H}\Pi^{(i)}\hat{\boldsymbol{b}}_l\|^2$, which reduces to

$$c_l = \left\|\mathbf{D}(\boldsymbol{y} - \mathbf{M}\hat{\boldsymbol{b}}_l)\right\|^2 \tag{23}$$

where $\hat{\boldsymbol{b}}_l$ is the decision vector produced by the lth subdetector. For the case when $\alpha^2 = N_0$, (23) becomes an approximation due to the residual ISI.

Two crucial means for reducing complexity deserve to be highlighted.

- The computations made inside the subdetectors can be reused to calculate the cost. Specifically, using (22) and the fact that $m_{k,k} = 1$, we can rewrite the cost expression (23) as

$$c_l = \sum_{k=1}^{N} d_{k,k}^2|x_{k,l} - \hat{b}_{k,l}|^2. \tag{24}$$

  Therefore, calculating the cost for a subdetector decision vector requires at most only $O(N)$ additional computations.
- A *pruning* and *threshold-tightening* strategy can be used to avoid unnecessary calculations. In particular, a cost *threshold* τ can be established with the cost c_1 of the first subdetector's decision. In subsequent subdetectors, we can abort both the cost calculation (24) as well as the decision feedback process (22) whenever this threshold is exceeded (see Line 9 of Fig. 4). Furthermore, the threshold can be

reduced each time a lower cost is found (see Line 15 of Fig. 4).

As presented here, the B-Chase algorithm implements the subdetectors in serial fashion. The B-Chase detector also lends itself to a parallel implementation since each of the subdetectors can operate independently, as portrayed in Fig. 3.

V. Numerical Results

This section examines the performance and complexity of B-Chase detectors on Rayleigh-fading channels, assuming the channel parameters $\mathbf{H}$ and N_0 are known to the receiver. We will compare the MMSE B-Chase detector to the ZF and MMSE sphere detectors as implemented in [6] whose initial radii are set to infinity. Setting the initial radius to infinity for these sphere detectors is equivalent to setting it to the mean-squared error of the output of the ZF and MMSE BODF detectors, respectively. That enables the ZF sphere detector to achieve ML performance. We also compare against the lattice-reduced MMSE BODF (LR-BODF) and lattice-reduced MMSE linear (LR-linear) detectors [21]. The last detector we compare against is the ML-DF [12] detector, which detects the first three symbols using ZF sphere detection [6], and the final symbol using ZF DF detection. We will first give numerical results for the performance and complexity of these detectors individually, then jointly. We use B-Chase$^\star(\ell)$ to denote the B-Chase detector with list length ℓ, $\alpha^2 = N_0$, and selection algorithm (14). Likewise, we use B-Chase(ℓ) to denote the B-Chase detector with list length ℓ, $\alpha^2 = N_0$, and selection algorithm (16). The MMSE versions of the parallel and BODF detectors are also included in the comparison, since they are the special cases B-Chase$^*(|A|)$ and B-Chase(1), respectively.

The B-Chase detector achieves near-ML performance for a variety of channel dimensions. To demonstrate this we performed simulations over N-input N-output Rayleigh-fading channels with 16-QAM inputs. Fig. 6 shows the performance versus the number of antennas, where the SNR per bit is $\mathrm{E}[||\mathbf{Ha}||^2]/(\mathrm{E}[||\boldsymbol{w}||^2] \cdot \log_2 |A|)$. We see that B-Chase(16) achieves near-ML performance, with an SNR penalty that ranges from 0.5 dB to 1.0 dB as the number of antennas N increases from 2 to 6. Reducing the list length degrades performance, but B-Chase(4) performs at least as well as the LR-BODF detector over the range of N from 2 to 6.

We now quantify the complexity of the B-Chase detector. The best complexity metric depends upon many variables that are specific to a particular implementation. We avoid the problem of defining the relative complexity of different floating-point operations by measuring complexity as the total number of real multiplies (RMs) per bit. The squared absolute value of a complex number is counted as two RM, and complex multiplications are counted as three RMs. Since the number of divisions and square-roots is small compared to the number of multiplies, the main drawback of counting only the multiplies is that it neglects the contribution to the complexity of the addition operations. However, this is a reasonable simplification since multiplies are generally more complex to implement than additions. Another important point is that the multiplication of a floating-point number by a constellation point is counted as an *addition* since the constellation points are just scaled integers [30]. This means that implementing interference cancellation (22) is multiply free.

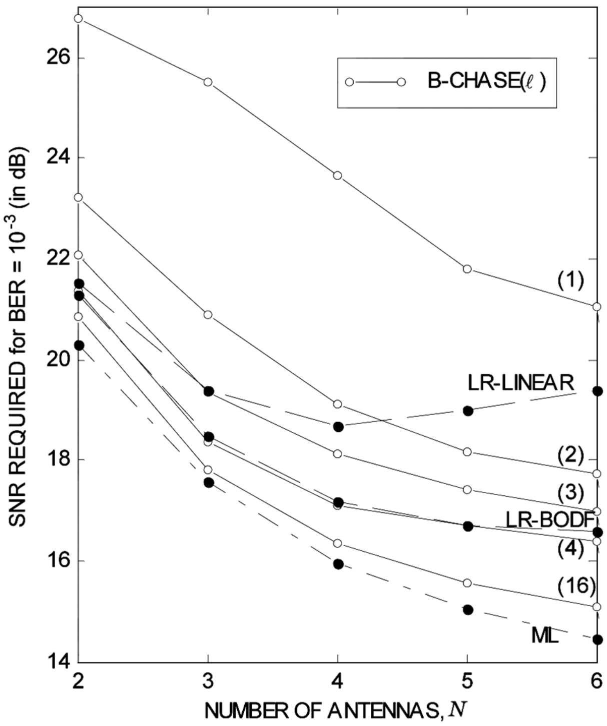


Fig. 6. SNR required versus number of antennas for various detectors. Results are averaged over 10^5 Rayleigh-fading $N \times N$ channels with 16-QAM inputs.

The number of computations required by the detectors we compare varies for different channel and noise realizations. Using the *average* complexity as the basis for comparison may be too optimistic, since systems are often designed to handle the worst-case scenario. On the other hand, the worst-case complexity may be too pessimistic since a practical system could enforce limits on complexity that are sufficiently high so as to have only a negligible effect on performance. One benefit of the B-Chase detector is that even in the worst case, it is still low in complexity. On the other hand, the worst-case complexity of the sphere detector and LLL algorithm can be extremely large. In order to give a fair and practical complexity comparison, we choose the complexity limit of the detector such that the probability that it is exceeded is the same as the target probability of a bit error. In other words, since the target BER is 10^{-3}, we quantify complexity using the 99.9% quantile of real multiplies.

The *preprocessing* complexity includes those computations that are required only once per channel estimation. The preprocessing used to implement the B-Chase* detector is described in Fig. 5, where the N sorted-QR decompositions dominate the preprocessing complexity. On the other hand, the most complex part of the preprocessing used to implement the B-Chase detector is the QR decomposition of the extended channel matrix in line 1 of Fig. 5. The preprocessing complexities of the MMSE sphere, LR-BODF, and LR-linear detectors are higher than that of the B-Chase detector. Although the preprocessing for the MMSE sphere detector is essentially the same as that of B-Chase(1), it is more complex because it uses the real channel model which doubles the channel dimensions. The LR-BODF detector requires the same preprocessing as the MMSE sphere detector in addition to LLL lattice reduction.

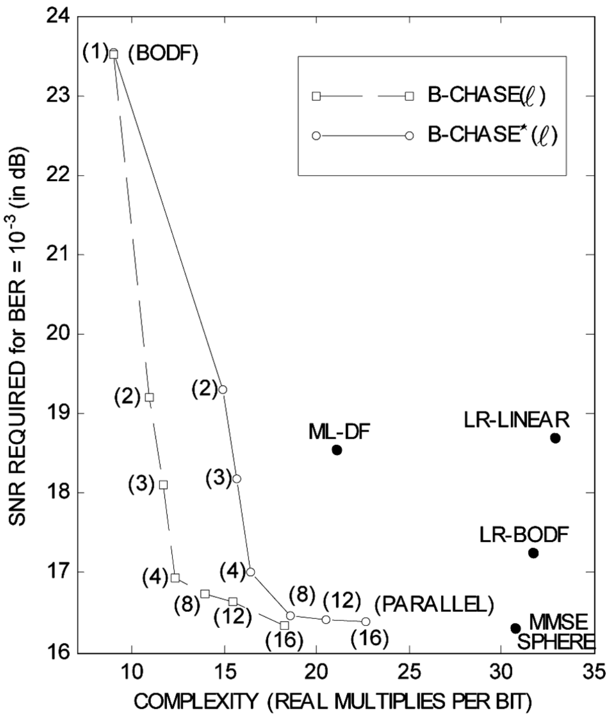


Fig. 7. Performance-complexity trade-off averaged over 10^5 Rayleigh-fading 4×4 channels with 16-QAM inputs, and $T = 8$.

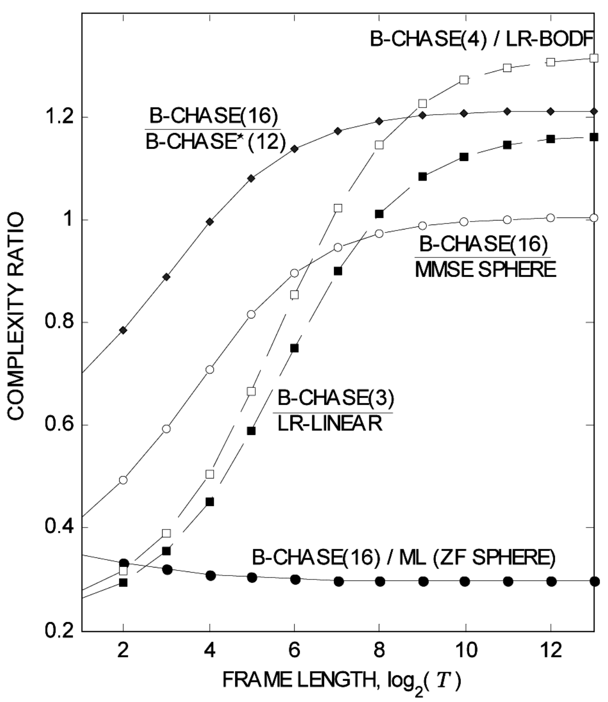


Fig. 8. Complexity ratio of various detectors averaged over 10^5 Rayleigh-fading 4×4 channels with 16-QAM inputs. Complexity is measured as 99.9% quantile of the total number of real multiplies required to reach BER $= 10^{-3}$.

We define the *core-processing* complexity as those computations which must be implemented during every symbol period. Fig. 4 describes the core-processing of both the B-Chase* and B-Chase detectors. When $\ell = 1$, it requires only $3MN$ RM since Lines 7 and 12 can be skipped. Otherwise it requires a maximum of $3(M + \ell)N$ RM. We assume that the channel estimate is updated every T symbol periods. As a result the *total complexity*, as measured by real multiples per bit, is related to the preprocessing complexity C_{pre} and core-processing complexity C_{core} by

$$\text{COMPLEXITY} = \frac{C_{\text{core}} + C_{\text{pre}}/T}{N \cdot \log_2 |A|}. \tag{25}$$

We now investigate the performance–complexity tradeoff of the B-Chase detectors for a four-input four-output Rayleigh-fading channel with 16-QAM inputs. Fig. 7 illustrates the performance versus complexity trade-off of each detector with a single point, where performance is measured by the SNR required to reach BER $= 10^{-3}$, and complexity is measured by the 99.9% quantile of the total real multiplies per bit (RM/ bit). The channel is assumed to change every eight symbol periods ($T = 8$). Not shown is the ML detector, which required 57 RM/b and 16.0 dB using the ZF sphere detector implementation. Also, it is worth noting that starting the sphere detectors from a noninfinite initial radius [9] decreased the average complexity, but increased the 99.9% quantile of complexity. B-Chase(16) sacrifices 0.4 dB of performance in order to reduce complexity by 68%, from 57 to 18 RM/b. At the low-complexity end of the spectrum, B-Chase(2) outperforms the BODF detector (B-Chase(1)) by 4.4 dB, while increasing the complexity by 17%, from 9 to 10.9 RM/b. B-Chase(16) not only outperforms the LR-BODF, LR-linear, and ML-DF detectors, but also reduces complexity by 42%, 44%, and 13%, respectively. B-Chase(16) falls only 0.1 dB short of the MMSE sphere detector, but required 41% fewer RM/b. The B-Chase* detector obtained relatively little performance improvement over the B-Chase detector. Clearly, for this scenario, the B-Chase detector exhibits a better performance–complexity tradeoff than the other low-complexity detectors. In addition, by simply adjusting the list length parameter, the B-Chase detector provides an effective way to trade complexity for performance.

An important dimension of the complexity comparison is not represented in Fig. 7 because it does not show the complexity comparison as a function of how quickly the channel changes. The relative complexity of the detectors depends upon how often the preprocessing is performed compared to the core processing. In order to demonstrate how T impacts detection complexity, Fig. 8 illustrates the complexity ratio between several pairs of detectors which have similar performance (see Fig. 7) versus T. First, B-Chase(16) performed within 0.1 dB of the MMSE sphere detector, while reducing complexity by as much as 62% when $T = 1$ and requiring practically the same complexity when $T > 512$. In a second comparison, B-Chase(3) outperformed the LR-linear detector by 0.5 dB, and was less complex for $T < 231$; reducing complexity by up to 76% when $T = 1$. Next, B-Chase(4) outperformed the LR-BODF detector by 0.1 dB, and was less complex for $T < 114$; reducing complexity by up to 74% when $T = 1$. Finally, B-Chase(16) performed the same as B-Chase*(12), and reduced complexity when $T \leq 16$. As T increases, the ability of the B-Chase* detector to reduce the list length with minimal performance loss outweighs the cost of its increased preprocessing complexity. These results show that the large investment in preprocessing made by the MMSE sphere, LR-BODF, LR-linear, and B-Chase* detectors does not pay off unless T is quite large.

VI. Conclusion

The Chase family of detection algorithms for MIMO channels is a combination of a list detector and a parallel bank of subdetectors. The general Chase detector reduces to a variety of existing MIMO detectors as special cases. Based on the Chase framework, we proposed the B-Chase detector that can trade performance for reduced complexity by modifying the list length. Using efficient implementations and a new selection algorithm, the B-Chase detector achieves near-ML performance with low complexity. For example, on a four-input four-output Rayleigh-fading channel that changes every eight symbol periods, and whose inputs are uncoded 16-QAM, the B-Chase(16) detector fell 0.4 dB short of the ML detector while reducing complexity by 68%. Compared to the MMSE sphere detector, the B-Chase(16) fell only 0.1 dB short while reducing complexity by 41%. At the low end of the complexity spectrum, the B-Chase(2) detector outperformed the MMSE BODF detector by 4.4 dB while increasing complexity by only 17%.

References

[1] G. J. Foschini, G. Golden, R. Valenzuela, and P. Wolniansky, "Simplified processing for wireless communication at high spectral efficiency," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[2] M. K. Varanasi, "Group detection for synchronous Gaussian code-division multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 41, no. 4, pp. 1083–1096, Jul. 1995.

[3] J. Luo, K. Pattipati, P. Willett, and G. Levchuk, "Optimal grouping for a group decision feedback detector in synchronous CDMA communications," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 341–346, Mar. 2003.

[4] E. Viterbo and E. Biglieri, "A universal lattice decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 2400–2414, Oct. 2003.

[5] A. Chan and I. Lee, "A new reduced-complexity sphere decoder for multiple antenna systems," in *Proc. IEEE Conf. Commun.*, 2002, pp. 460–464.

[6] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[7] E. Zimmerman, W. Rave, and G. Fettweis, "On the complexity of sphere decoding," presented at the Int. Symp. Wireless and Pers. Multimedia Commun. (WPMC), Abano Terme, Italy, Sep. 2004.

[8] K. Su and I. J. Wassell, "A new ordering for efficient sphere decoding," *Proc. IEEE Int. Conf. Commun.*, vol. 3, pp. 1906–1910, May 2005.

[9] W. Zhao and G. B. Giannakis, "Sphere decoding algorithms with improved radius search," in *Proc. IEEE Commun. Networking Conf.*, Mar. 2004, vol. 4, pp. 2290–2294.

[10] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[11] D. Pham, K. R. Pattipati, P. K. Willett, and J. Luo, "An improved complex sphere decoder for V-BLAST systems," *IEEE Signal Process. Lett.*, vol. 11, no. 9, pp. 748–751, Sep. 2004.

[12] W. J. Choi, R. Negi, and J. Cioffi, "Combined ML and DFE decoding for the V-BLAST system," in *Proc. IEEE Conf. Commun.*, Jun. 2000, pp. 1243–1248.

[13] A. Bhargave, R. J. P. de Figueiredo, and T. Eltoft, "A detection algorithm for the V-BLAST system," in *Proc. IEEE Global Telecommun. Conf. (IEEE GLOBECOM)*, Nov. 2001, vol. 1, pp. 494–498.

[14] F. Tu, D. Pham, J. Luo, K. Pattipati, and P. Willett, "Decision feedback with rollout for multiuser detection in synchronous CDMA," *Proc. Inst. Electr. Eng.—Commun.*, vol. 151, no. 4, pp. 383–386, Aug. 2004.

[15] Y. Li and Z. Luo, "Parallel detection for V-BLAST system," in *Proc. IEEE Conf. Commun.*, May 2002, vol. 1, pp. 340–344.

[16] H. Sung, K. B. Lee, and J. W. Kang, "A simplified maximum likelihood detection scheme for MIMO systems," in *Proc. IEEE Vehicular Technol. Conf.*, Oct. 2003, vol. 1, pp. 419–423.

[17] M. Rupp, G. Gritsch, and H. Weinrichter, "Approximate ML detection for MIMO systems with very low complexity," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, May 2004, vol. 4, pp. 809–812.

[18] J. H.-Y. Fan, R. D. Murch, and W. H. Mow, "Near maximum likelihood detection schemes for wireless MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 3, no. 5, pp. 1427–1430, Sep. 2004.

[19] H. Yao and G. W. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," in *Proc. Global Telecommun. Conf. (IEEE GLOBECOM)*, Nov. 2002, vol. 1, pp. 424–428.

[20] C. Windpassinger and R. F. H. Fischer, "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Apr. 2003, pp. 345–348.

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

[22] C. Windpassinger, L. H.-J. Lampe, and R. F. H. Fischer, "From lattice-reduction-aided detection towards maximum-likelihood detection in MIMO systems," in *Proc. Int. Conf. Wireless Optical Commun. (WOC)*, Jul. 2003, pp. 144–148.

[23] J. Jaldén, L. G. Barbero, B. Ottersten, and J. S. Thompson, "Full diversity detection in MIMO systems with a fixed-complexity sphere decoder," in *Proc. IEEE Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Apr. 2007, vol. 3, pp. 49–52.

[24] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. 18, no. 1, pp. 170–182, Jan. 1972.

[25] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE Conf. Acoustics, Speech, Signal Processing*, Jun. 2000, vol. 2, pp. 737–740.

[26] R. Böhnke, D. Wübben, V. Kühn, and K. Kammeyer, "Reduced complexity MMSE detection for BLAST architectures," in *Proc. IEEE Global Telecommun. Conf. (IEEE GLOBECOM)*, Dec. 2003, vol. 4, pp. 2258–2262.

[27] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *Electron. Lett.*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

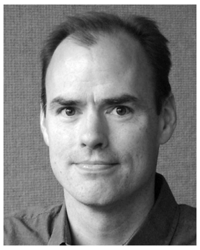
[28] D. W. Waters and J. R. Barry, "Noise-predictive decision-feedback detection for multiple-input multiple-output channels," *IEEE Trans. Signal Process.*, vol. 53, no. 5, pp. 1852–1859, May 2005.

[29] A. Duel-Hallen, "Decorrelating decision-feedback multiuser detector for synchronous code-division multiple access channel," *IEEE Trans. Commun.*, vol. 41, no. 2, pp. 285–290, Feb. 1993.

[30] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

Deric W. Waters (S'99–M'02) was born in Wellington, TX, in 1977. He received the B.S. degrees in electrical engineering and computer science from Texas Tech University, Lubbock, in 1999, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 2002 and 2005, respectively.

Currently, he is a System Engineer in the Digital Signal Processing and Systems Research and Development Laboratory of Texas Instruments, Dallas, TX.



John R. Barry (S'85–M'87–SM'04) received the B.S. degree in electrical engineering from the State University of New York, Buffalo, in 1986 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1987 and 1992, respectively.

Since 1992, he has been with the Georgia Institute of Technology, Atlanta, where he is currently a Professor with the School of Electrical and Computer Engineering. His research interests include wireless communications, equalization, and multiuser communications. He is coauthor with E. A. Lee and D. G. Messerschmitt of *Digital Communications* (Norwell, MA: Kluwer, 2004, 3rd ed.) and the author of *Wireless Infrared Communications* (Norwell, MA: Kluwer, 1994).