A General MIMO Detection Scheme and Its Performance-Complexity Tradeoff

Ronald Y. Chang and Wei-Ho Chung*
Research Center for Information Technology Innovation, Academia Sinica, Taiwan

Abstract— **A unified tree-search detection scheme based on Dijkstra's algorithm is developed for MIMO systems. The proposed framework generalizes the original Dijkstra's algorithm by allowing the memory usage, detection complexity, and sorting dynamic associated with the algorithm to be customized. By tuning different parameters, desired performance-complexity tradeoffs are attained and a fixed-complexity version can be produced to facilitate hardware implementation. Simulation results demonstrate that the proposed algorithm shows abilities to achieve highly favorable performance-complexity tradeoffs.**

I. Introduction

The multiple-input multiple-output (MIMO) detection problem can be viewed as a tree-search problem, where optimal detection corresponds to finding the leaf node with the smallest cost of likelihood metric [1]. Since exhaustive search of the tree is computationally prohibitive, various optimal and suboptimal methods with reduced complexity have been proposed. These methods, categorized in tree-search terms, include depth-first search (sphere decoding or SD [2] [3]), breadth-first search (K-best detection [4] [5]), and best-first search (the stack or Dijkstra's algorithm [6]–[8]). Hybrid schemes were also proposed to leverage the advantages of different methods, e.g., the combination of SD and Dijkstra's algorithm in [9] [10]. Furthermore, unification frameworks building on relations between different categories were suggested, including the generic branch and bound algorithm in [6] and the unified algorithm in [10] which augments the depth-first search scheme with the best-first ability to improve search complexity.

Unification is attractive from an algorithmic as well as a performance point of view, as it provides a general framework that incorporates strengths of different schemes. The idea of this work is to extend the original Dijkstra's algorithm to a generalized framework for MIMO detection. Our approach differs from the unification in [6] [10] in the following aspects:

1) *Memory usage:* In order to implement Dijkstra's algorithm in practice, its extensive memory requirements have to be addressed. Different from the memory-efficient modifications in [6] [10], our scheme adopts an explicit memory constraint and nodes are visited only once, similar to [8].

*Corresponding author: Wei-Ho Chung. Email: whc@citi.sinica.edu.tw. This work was supported in part by National Science Council of Taiwan under grant number NSC 100-2221-E-001-004, and Industrial Technology Research Institute of Taiwan under MOEA project.

2) *Sorting scheme:* Sorting in Dijkstra's algorithm determines tree exploration and the performance of the algorithm. As reported in our previous work [11], manipulating sorting dynamics can yield better performance-complexity tradeoffs given the same memory constraint. Different from the unification in [6] [10], our scheme accommodates adaptable sorting that is designed specifically from the statistics of the problem.

The proposed generalized Dijkstra's algorithm incorporates a parameter triplet that are used to control the memory usage, expansion complexity, and sorting dynamic associated with the algorithm. The scheme, referred to as generalized Dijkstra's search (GDS) algorithm, is demonstrated by simulation to show flexibility in achieving improved performance-complexity tradeoffs.

The rest of the paper is organized as follows. In Sec. II, the system model is presented and a tree-search detection scheme is reviewed. Our proposed method is described in Sec. III with its performance demonstrated in Sec. IV. Finally, concluding remarks are given in Sec. V.

II. System Model and Tree-Search Detection

A. System Model and Problem Description

We consider an uncoded MIMO transmission system with N_T transmit antennas and N_R receive antennas ($N_R \geq N_T$). The baseband signal model is given by

$$\mathbf{y}_c = \mathbf{H}_c \tilde{\mathbf{x}}_c + \mathbf{v}_c, \tag{1}$$

where $\mathbf{y}_c$ is the $N_R \times 1$ received signal composed of the $N_T \times 1$ transmitted signal $\tilde{\mathbf{x}}_c$ passed through the $N_R \times N_T$ flat-fading channel $\mathbf{H}_c$ plus the $N_R \times 1$ perturbing noise vector $\mathbf{v}_c$. The transmitted symbol vector $\tilde{\mathbf{x}}_c$ contains uncorrelated entries drawn equally probably from the squared quadrature amplitude modulation (QAM) alphabet $\mathbb{S} = \{a + ib \mid a, b \in \mathbb{Q}\}$, where $\mathbb{Q}$ is the pulse amplitude modulation (PAM) alphabet, and has zero mean and covariance matrix $\sigma_x^2 \mathbf{I}_{N_T}$, where $\mathbf{I}_{N_T}$ is the $N_T \times N_T$ identity matrix. The complex-valued channel matrix $\mathbf{H}_c$ has independent and identically distributed (i.i.d.) Gaussian entries with zero mean and covariance matrix $\sigma_H^2 \mathbf{I}_{N_R}$, where $\sigma_H^2 = 1$. The channel information $\mathbf{H}_c$ is assumed perfectly known to the receiver. The noise $\mathbf{v}_c$ is additive white Gaussian noise (AWGN) with i.i.d. complex elements and has zero mean and covariance matrix $\sigma_v^2 \mathbf{I}_{N_R}$.

For the discussion in the subsequent sections, it is convenient to consider an equivalent real signal model to the complex signal model in (1). Denote $\Re(\cdot)$ and $\Im(\cdot)$ as the real and imaginary parts of its argument, respectively, and define

$$\mathbf{y} = \begin{bmatrix} \Re(\mathbf{y}_c) \\ \Im(\mathbf{y}_c) \end{bmatrix}, \tilde{\mathbf{x}} = \begin{bmatrix} \Re(\tilde{\mathbf{x}}_c) \\ \Im(\tilde{\mathbf{x}}_c) \end{bmatrix}, \mathbf{v} = \begin{bmatrix} \Re(\mathbf{v}_c) \\ \Im(\mathbf{v}_c) \end{bmatrix},$$
$$\mathbf{H} = \begin{bmatrix} \Re(\mathbf{H}_c) & -\Im(\mathbf{H}_c) \\ \Im(\mathbf{H}_c) & \Re(\mathbf{H}_c) \end{bmatrix}.$$

Then, the real signal model is given by

$$\mathbf{y} = \mathbf{H}\tilde{\mathbf{x}} + \mathbf{v}, \tag{2}$$

where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{H} \in \mathbb{R}^{n\times m}$, $\tilde{\mathbf{x}} \in \mathbb{Q}^m$, $\mathbf{v} \in \mathbb{R}^n$ with $n = 2N_R$ and $m = 2N_T$. Given the signal model in (2), the optimal ML detection is equivalent to solving a constrained least-square problem, i.e.,

$$\tilde{\mathbf{x}}_{\text{ML}} = \arg\min_{\mathbf{x}\in\mathbb{Q}^m} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2, \tag{3}$$

where $\|\cdot\|$ denotes the l_2-norm of a vector.

B. Dijkstra's Search (DS) Algorithm for Tree-Search Detection

The detection problem in (3) has a tree representation [1]. (For the simplicity of presentation, we hereafter assume $m = n$.) This can be seen by first performing QR decomposition on $\mathbf{H}$ such that $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper-triangular matrix with positive-valued diagonal entries (assuming $\mathbf{H}$ has full rank), and then formulating ML detection into an equivalent expression $\tilde{\mathbf{x}}_{\text{ML}} = \arg\min_{\mathbf{x}\in\mathbb{Q}^m} \|\mathbf{y}' - \mathbf{R}\mathbf{x}\|^2$, where $\mathbf{y}' = \mathbf{Q}^T\mathbf{y}$ and $(\cdot)^T$ denotes the matrix transpose. Due to the upper-triangular nature of $\mathbf{R}$, $\|\mathbf{y}' - \mathbf{R}\mathbf{x}\|^2$ can be expanded in the following form:

$$(y'_m - r_{m,m}x_m)^2 + \left(y'_{m-1} - \sum_{i=m-1}^{m} r_{m-1,i}x_i\right)^2 + \cdots$$
$$+ \left(y'_1 - \sum_{i=1}^{m} r_{1,i}x_i\right)^2, \tag{4}$$

where y'_i is the ith element of $\mathbf{y}'$, x_i is the ith element of $\mathbf{x}$, and $r_{i,j}$ is the (i,j)-entry of $\mathbf{R}$. Denote the $(m-k+1)$th term in (4) by $B_k(\mathbf{x}_k^m)$ and the summation of the first $m-k+1$ terms by $D_k(\mathbf{x}_k^m)$, where $\mathbf{x}_k^m \triangleq (x_k, \ldots, x_m)^T \in \mathbb{Q}^{m-k+1}$ represents the partial symbol vector, i.e.,

$$B_k(\mathbf{x}_k^m) = \left(y'_k - \sum_{i=k}^{m} r_{k,i}x_i\right)^2, k = 1, 2, \ldots, m, \tag{5}$$
$$D_k(\mathbf{x}_k^m) = \sum_{j=k}^{m} B_j(\mathbf{x}_j^m), k = 1, 2, \ldots, m. \tag{6}$$

Then, the upper-triangular structure of $\mathbf{R}$ creates the detection tree, which consists of a virtual root node and m layers of nodes where each non-leaf node has $|\mathbb{Q}|$ child nodes, $|\cdot|$ denoting the cardinality of a set. Each node in layer $m-k+1$ ($k = 1, 2, \ldots, m$) uniquely represents an $\mathbf{x}_k^m$ and has an associated *path metric* $D_k(\mathbf{x}_k^m)$ and *branch metric* $B_k(\mathbf{x}_k^m)$. In particular, $D_1(\mathbf{x}_1^m)$ of a leaf node in layer m equals $\|\mathbf{y}' - \mathbf{R}\mathbf{x}\|^2$ evaluated for the particular $\mathbf{x} = \mathbf{x}_1^m$ represented

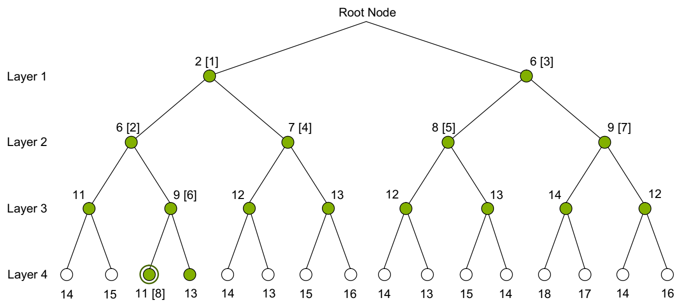


Fig. 1. An example of MIMO detection based on the DS algorithm for a 2×2 system with 4-QAM modulation ($m = 4$ and $|\mathbb{Q}| = 2$). The number labeled next to a node is its path metric, and the number in the brackets is the order in which the node is selected as the best node and thus expanded. Colored nodes represent the total visited nodes. The circled leaf node is the optimal (ML) solution.

by the node. Hence, the objective of optimal detection is to find the leaf node with the smallest path metric among all leaf nodes.

Dijkstra's algorithm for finding the shortest path in a graph can be applied to solving the MIMO detection problem [7]. It maintains a list of nodes sorted in ascending order of their path metric and explores the nodes in such order. The algorithm is described as follows.

Algorithm: DS

Initially, the node list contains only the root node.

1) *(Selecting the best node)* Select the first node (the *best node* in this iteration) from the node list. If this node is in layer m (i.e., a leaf node), stop the algorithm and output it as the solution.
2) *(Expanding the best node)* Expand the best node by adding all its children nodes to the node list and removing itself from it.
3) *(Sorting the node list)* Order the nodes in the node list in ascending order of their path metric.[1] Go to step 1.

An illustrative example of applying the DS algorithm to solving the detection problem is shown in Fig. 1, where the algorithm iterations are depicted. Visited nodes are defined by those that ever occupy a position in the node list.

III. The Proposed Generalized Dijkstra's Search (GDS) Algorithm

The GDS algorithm generalizes the DS algorithm by introducing three configurable input parameters through which the memory usage, expansion complexity, and sorting dynamic associated with the algorithm can be customized. As a result, the GDS algorithm includes some existing schemes as special cases. In the following, the algorithm is first presented and then its properties are discussed.

A. Algorithm Description

Before we present the proposed algorithm, we first introduce two relevant results.

[1]Ties are broken by ordering nodes in higher layers (closer to the bottom of the tree) before nodes in lower layers, but otherwise arbitrarily.

1) Statistics of the path metric: In the tree representation, in each layer k there is exactly one node that corresponds to the actually transmitted $\tilde{\mathbf{x}}_{m-k+1}^{m}$. The path metric for this particular node, $D_{m-k+1}(\tilde{\mathbf{x}}_{m-k+1}^{m})$, is the summation of the square of k random variables that are i.i.d. $\mathcal{N}(0, \sigma_v^2/2)$ and is distributed according to the chi-square distribution with k degrees of freedom. Its cumulative distribution function (cdf) is given by

$$F(x;k) = \frac{\gamma(k/2, x/\sigma_v^2)}{\Gamma(k/2)}, x \geq 0, k = 1, 2, \ldots, m, \quad (7)$$

where $\Gamma(\cdot)$ is the Gamma function and $\gamma(\cdot)$ is the incomplete Gamma function [12]. Let F^{-1} be the inverse function of F, such that $x = F^{-1}(y;k)$ if $y = F(x;k)$.

2) Proposed generalized path metric: As a best-first tree-search method, the GDS algorithm maintains an ordered node list during the tree-search process. In the DS algorithm, ordering is conducted according to nodes' path metric, oblivious of the layer information of nodes. The Dijkstra's search with probabilistic sorting (DSPS) algorithm recently developed in [11] consults the statistics of the path metric to order nodes of different layers on a fairer basis to enhance the DS algorithm. More specifically, nodes are ordered by their *normalized path metric*, defined as $G_N(\mathbf{x}_{m-k+1}^{m}) \triangleq F(D_{m-k+1}(\mathbf{x}_{m-k+1}^{m}); k)$ for a node $\mathbf{x}_{m-k+1}^{m}$ in layer k.[2] One problem with this sorting scheme is that, albeit achieving reduced complexity, the detection algorithm that employs this sorting scheme does not always find the ML solution. We are therefore motivated to develop a new *generalized path metric* which will be shown to remove such a limitation. The generalized path metric for a node $\mathbf{x}_{m-k+1}^{m}$ in layer k is defined as

$$G^{(T)}(\mathbf{x}_{m-k+1}^{m}) \triangleq F^{-1}\Big(F\big(D_{m-k+1}(\mathbf{x}_{m-k+1}^{m}); k\big); k+T\Big), \quad (8)$$

where $T \in \{0, \mathbb{Z}^+\}$ is a parameter and $\mathbb{Z}^+$ is the set of positive integers. The special case of $T = 0$ returns the path metric $D_{m-k+1}(\mathbf{x}_{m-k+1}^{m})$.

The proposed GDS algorithm introduces three parameters $L \in \mathbb{Z}^+$, $K \in \mathbb{Z}^+$, and T through which the memory usage, expansion complexity, and sorting dynamic associated with the algorithm are configured, respectively. The GDS algorithm is described as follows.

```
Algorithm: GDS($L$, $K$, $T$)
    Initially, the node list $\mathbb{C} = \{N_{(1)}, N_{(2)}, \ldots\}$ contains
    only the root node, as $N_{(1)}$.
    1) (Selecting the leading nodes) Select the leading nodes
       $N_{(1)}, N_{(2)}, \ldots, N_{(\min(|\mathbb{C}|, K))}$ from $\mathbb{C}$. If $N_{(1)}$ is in
       layer $m$ (i.e., a leaf node), stop the algorithm and
       output it as the solution.
    2) (Expanding the leading nodes) Expand each of the
       leading nodes $N_{(i)}, i = 1, 2, \ldots, \min(|\mathbb{C}|, K)$, by
       adding its children nodes to $\mathbb{C}$ and removing itself
       from $\mathbb{C}$. If any of the $\min(|\mathbb{C}|, K)$ leading nodes is
       in layer $m$, leave it in $\mathbb{C}$ and do not expand it.
    3) (Maintaining and sorting the node list) Sort the nodes
       in $\mathbb{C}$ in ascending order of their generalized path metric
       $G^{(T)}$. Retain $N_{(1)}, N_{(2)}, \ldots, N_{(\min(|\mathbb{C}|, L))}$ and discard
       others. Go to step 1.
```

[2]It is termed *normalized* because a node's path metric relative to the distribution of the path metric of the transmitted partial symbol vector in the same layer is considered. See [11] for more details.

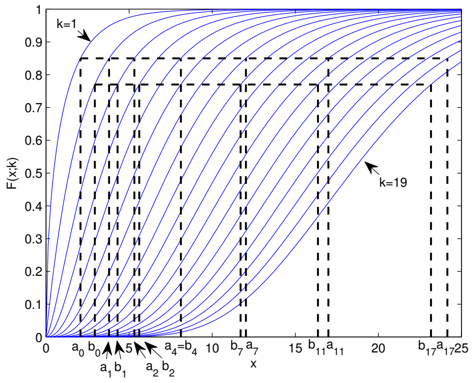


Fig. 2. Plot of $F(x;k)$ in (7) for $k = 1, 2, \ldots, 19$ ($\sigma_v^2 = 2$), with the generalized path metric a_T and b_T of two example nodes shown for $T = 0, 1, 2, 4, 7, 11,$ and 17.

B. Properties

The GDS algorithm takes in the parameter triplet (L, K, T) and thereby offers rich configurability. Its properties are discussed as follows.

1) Memory usage: The traditional challenge of the extensive memory usage in Dijkstra's algorithm is addressed by the introduction of L, similar to [8]. In fact, given some L and K, the GDS algorithm requires a fixed memory size of $L + K(|\mathbb{Q}| - 1)$, each unit for storing the (partial) symbol vector and the path metric. Compared to the SD algorithm that requires a fixed memory size of $(m-1)|\mathbb{Q}| + 1$ [9], the GDS algorithm is comparably memory-efficient, as moderate L and $K = 1$ are sufficient to yield near-optimal performance, as presented in Sec. IV.

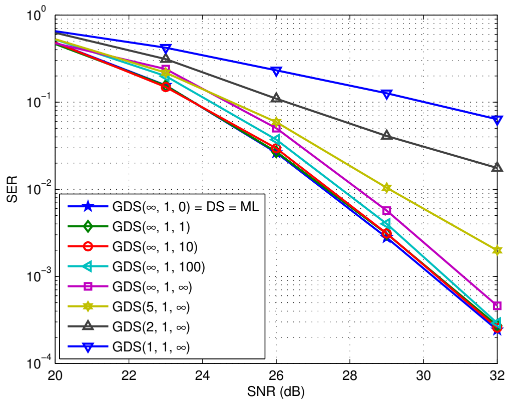
2) Fixing the complexity: The parameter K determines the expansion complexity. The original Dijkstra's algorithm employs $K = 1$. Increasing K will increase the expansion complexity, yet may or may not increase the performance (see Sec. IV). The main purpose of introducing this parameter is to fix the complexity of GDS. In fact, by setting $L = K$ with an arbitrary T, GDS becomes equivalent to a K-best detector. In this setting, the detection proceeds uniformly towards the bottom of the tree and the GDS algorithm stops after a fixed number of $m = 2N_T$ iterations for an $N_T \times N_T$ system, thus producing fixed throughput and facilitating efficient pipelined implementation.

3) Sorting dynamic: The parameter T controls the sorting dynamic, through the generalized path metric in (8). The effect that different values of T produce in sorting is illustrated in Fig. 2. Consider two exemplary nodes, $\mathbf{x}_m^m$ in layer 1 and $\mathbf{x}_{m-1}^m$ in layer 2, with generalized path metric denoted by $a_T = G^{(T)}(\mathbf{x}_m^m)$ and $b_T = G^{(T)}(\mathbf{x}_{m-1}^m)$, respectively. It is seen in Fig. 2 that the relative magnitude of these two nodes' generalized path metric depends on T: for small T (e.g., $T = 0, 1, 2$), $a_T < b_T$, with diminishing differences as T increases; for some intermediate T (e.g., $T = 4$), they are nearly identical; and for larger T (e.g., $T = 7, 11, 17$), $a_T > b_T$. In fact, it is straightforward to see that sorting by $G^{(T)}$ reduces to sorting by the path metric when $T = 0$, and it is empirically shown that sorting by $G^{(T)}$ becomes equivalent to sorting by the normalized path metric as $T \to \infty$. An intermediate value of T, as can be inferred from Fig. 2, will produce a sorting dynamic in between the two extreme cases of $T = 0$ and $T = \infty$. Thus, it is reasonable to expect that choosing an intermediate-valued T will give a performance-complexity tradeoff between what can be attained by $T = 0$ and $T = \infty$ when other parameters are fixed. This conjecture is verified by simulation in the next section.

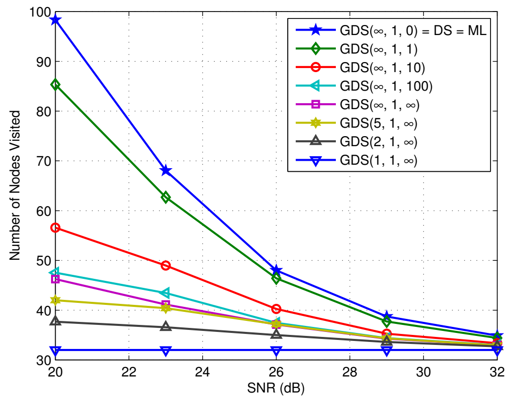
IV. Simulation Results and Discussions

In this section, we study the performance of the proposed GDS algorithm by computer simulation. The real signal model in (2) is implemented for uncoded MIMO systems. First, we plot the symbol error rate (SER) and complexity (in terms of the average number of nodes visited) results for different signal-to-noise ratios (SNRs) for a 4×4 system with 16-QAM modulation, in Fig. 3. It is seen that by fixing K and varying L and T we can strike flexible performance and complexity tradeoffs. Specifically, from the ML-achieving GDS(∞, 1, 0), which corresponds to the DS algorithm, increasing T from 0 to ∞ and further decreasing L from ∞ to 1 increasingly trades the performance for lower complexity in the resulting scheme.

To demonstrate the performance-complexity tradeoffs more explicitly and examine the effect of different parameter combinations, we plot the SER against the complexity at fixed SNR for two scenarios: 4×4 system with 16-QAM modulation (Fig. 4(a)) and 8×8 system with 4-QAM modulation (Fig. 4(b)). The modified Dijkstra's algorithm [8], the DSPS algorithm [11], and the probabilistic tree pruning SD (PTP-SD) scheme [3] as a configurable SD algorithm (with the Schnorr-Euchner (SE) enumeration [3] and an infinite initial sphere radius) are considered in our comparison. In both figures, the numbers labeled next to a line represent its parameter. It is seen in both figures that, while the modified Dijkstra's algorithm (the GDS(L, 1, 0) line) can produce different points, it is unable to produce points in the lower-left region of the plot, which are however attainable by the GDS algorithm through a joint configuration of memory usage (L) and sorting (T), as seen by the GDS(L, 1, ∞) and GDS(∞, 1, T) lines. In fact, interestingly, while both the GDS algorithm and the modified Dijkstra's algorithm attain the same end-points (zero-forcing decision-feedback-equalization (ZF-DFE) point at one end and the optimal point at the other), the GDS algorithm produces different intermediate points. Specifically, as the modified Dijkstra's algorithm goes from GDS(1, 1, 0) to GDS(∞, 1, 0), the GDS algorithm goes from GDS(1, 1, ∞) to GDS(∞, 1, ∞) to GDS(∞, 1, 0) in the lower-left region. Note that the



(a)



(b)

Fig. 3. Performance and complexity of MIMO detection schemes for a 4×4 system with 16-QAM modulation. (a) SER. (b) Complexity.

line section between GDS(∞, 1, ∞) and GDS(∞, 1, 0) is not achievable by DSPS but by GDS. By jointly designing L and T in the GDS algorithm, various desired tradeoff points can be produced.

In the heuristic performance regime, the GDS algorithm can potentially achieve lower complexity and higher performance than the modified Dijkstra's algorithm for the same L, e.g., GDS(5, 1, ∞) vs. GDS(5, 1, 0). Similar results are also observed in the near-optimal performance regime, e.g., along the line GDS(10, 1, T). Furthermore, in the heuristic regime, GDS(L, 1, ∞) exhibits a steeper slope than the modified Dijkstra's algorithm (and PTP-SD, too), indicating that the number of nodes visited increases more moderately as the performance improves as far as an order of magnitude.

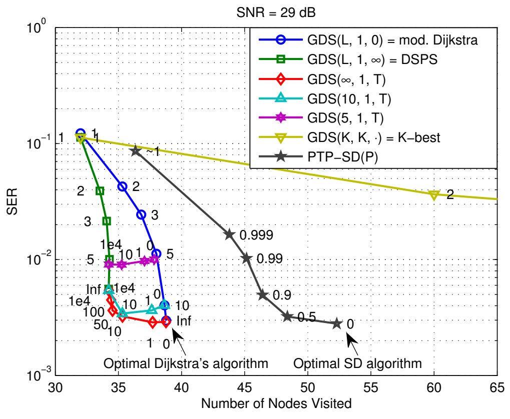
By setting $L = K$ with an arbitrary T (say, zero), the GDS algorithm becomes the K-best scheme as indicated by the GDS(K, K, $\cdot$) line. It is seen that such a configuration generally searches more nodes than needed to attain a certain SER performance, a price paid for the benefit of fixed complexity.

The effect of K on the performance is revealed by comparing GDS(K, 1, 0) and GDS(K, K, 0). As seen in Fig. 4(b), for small values of L, increasing K improves the performance at some cost of complexity (comparing GDS(3, 1, 0) and GDS(3, 3, 0), for example). As L increases, the performance gain from a greater K diminishes, and even parishes in the case of $L = \infty$, as both GDS(∞, 1, 0) and GDS(∞, ∞, 0) achieve the optimal performance.

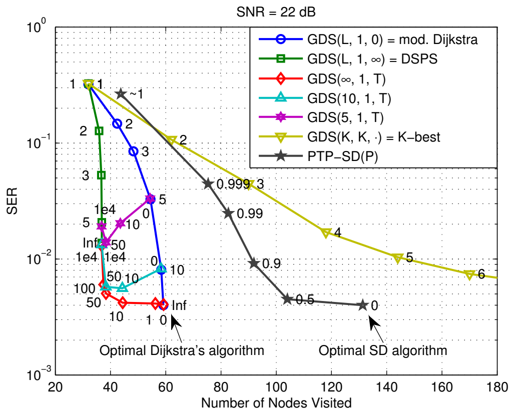
It should be emphasized that some computational costs (such as evaluating F and F^{-1}) are not accounted for in the comparison in Figs. 4(a)–4(b). However, we believe the number of searched nodes is a meaningful and important abstraction of the actual complexity involved to distinguish between different schemes which would otherwise be difficult to compare side-by-side. Besides, the results present a theoretical interest, showing that a more desired tradeoff region can be achieved. To reduce the computational cost of evaluating the cdf function and its inverse, closed-form approximations [13] and implementation of table lookups [14] are suggested for an online operation of the proposed algorithm.

V. Conclusion

A general Dijkstra's algorithm-based detection scheme for MIMO systems has been presented. The proposed theoretical framework enhances the original Dijkstra's algorithm by incorporating a configurable memory size, expansion complexity, and statistical information-based sorting mechanism. Computer simulation demonstrated that the proposed scheme presents noticeable advantages in terms of performance-complexity tradeoff as compared with the conventional Dijkstra's search scheme and the SD algorithm.



(a)



(b)

Fig. 4. Performance-complexity tradeoffs for MIMO detection schemes. (a) 4×4 system with 16-QAM modulation. (b) 8×8 system with 4-QAM modulation.

References

[1] E. G. Larsson, "MIMO detection methods: How they work," *IEEE Signal Processing Mag.*, vol. 26, pp. 91–95, May 2009.

[2] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Processing*, vol. 53, pp. 2806–2818, Aug. 2005.

[3] B. Shim and I. Kang, "Sphere decoding with a probabilistic tree pruning," *IEEE Trans. Signal Processing*, vol. 56, pp. 4867–4878, Oct. 2008.

[4] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2002, pp. III–273–III–276.

[5] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Select. Areas Commun.*, vol. 24, pp. 491–503, Mar. 2006.

[6] A. D. Murugan, H. El Gamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: Rediscovering the sequential decoder," *IEEE Trans. Inform. Theory*, vol. 52, pp. 933–953, Mar. 2006.

[7] K. Su, "Efficient maximum likelihood detection for communication over multiple input multiple output channels," *Ph.D. Dissertation*, University of Cambridge, 2005.

[8] T.-H. Kim and I.-C. Park, "High-throughput and area-efficient MIMO symbol detection based on modified Dijkstra's search," *IEEE Trans. Circuits Syst. I*, vol. 57, pp. 1756–1766, July 2010.

[9] Y. Dai and Z. Yan, "Memory-constrained tree search detection and new ordering schemes," *IEEE J. Select. Topics in Signal Processing*, vol. 3, pp. 1026–1037, Dec. 2009.

[10] C. Studer, A. Burg, and W. Fichtner, "A unification of ML-optimal tree-search decoders," in *Proc. 40th Asilomar Conference on Signals, Systems and Computers*, Oct. 2006, pp. 2185–2189.

[11] R. Y. Chang and W.-H. Chung, "Efficient tree-search MIMO detection with probabilistic node ordering," in *Proc. IEEE ICC '11*, June 2011.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. New York, NY: Cambridge University Press, 2007.

[13] M. Sankaran, "Approximations to the noncentral chi-square distribution," *Biometrika*, vol. 50, pp. 199–204, 1963.

[14] H. L. Harter and D. B. Owen, *Selected Tables in Mathematical Statistics. Volume 1*. Chicago: Markham Publishing Company, 1970.