Ph.D. Dissertation

Near-Optimal Signal Detection for Large MIMO Systems

by
Nguyen Thanh Nhan

August 2020

The Graduate School
Seoul National University of Science and Technology

Near-Optimal Signal Detection for Large MIMO Systems

by
Nguyen Thanh Nhan

A Ph.D. Dissertation submitted to the Department of Electrical and Information Engineering at the Graduate School of Seoul National University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy

August 2020

Approved by

Prof. Kyungchun Lee
Advisor

This certifies that the Ph.D. dissertation of
Nguyen Thanh Nhan is approved.

Committee Chairman: Prof. Taehyun Jeon

Committee Member: Prof. Kyungchun Lee

Committee Member: Prof. Dong-ho Kim

Committee Member: Prof. Ji-Hoon Yun

Committee Member: Prof. Illsoo Sohn

The Graduate School
Seoul National University of Science and Technology

2020 August

Summary

Title: Near-Optimal Signal Detection for Large MIMO Systems

In this thesis, we study the low-complexity near-optimal detection, which is an important challenge in optimizing practical multiuser massive multiple-input multiple-output (MIMO) systems. By optimizing the tabu search (TS) and sphere decoding (SD) schemes, we propose five novel near-optimal reduced-complexity detection algorithms, namely, the QR-decomposition-aided TS (QR-TS), neighbor-grouped TS (NG-TS), deep learning (DL)-aided TS (DL-TS), fast DL-aided SD (FDL-SD), and fast DL-aided $K-$best SD (KSD) (FDL-KSD), for large MIMO systems.

In the conventional TS detection algorithm for MIMO systems, the cost metrics of all neighboring vectors are computed to determine the best neighbor. This can require an excessively high computational complexity, especially in large MIMO systems because the number of neighboring vectors and the dimension per vector are large. The proposed QR-TS scheme allows for finding the best neighbor without computing all the neighbors' metrics by early-rejecting unpromising neighbors. By contrast, the NG-TS algorithm divides the neighbors into groups and finds the best neighbor by using a simplified cost function. To further optimize the QR-TS and NG-TS algorithms, novel ordering schemes are proposed. Simulation results show that QR-TS achieves a complexity reduction of approximately 60% – 90% with respect to (w.r.t.) the conventional TS scheme, while that attained by the NG-TS is up to 85%, without any performance loss.

In the proposed DL-aided detection schemes, namely, DL-TS, FDL-SD, and FDL-KSD, the proposed fast-convergence sparsely connected detection network (FS-Net) is incorporated with the TS, SD, and KSD schemes, respectively. In the DL-TS algorithm, the initial solution is approximated by the proposed FS-Net, and an adaptive early termination (ET) algorithm and a modified searching process are performed so that the optimal solution can be reached earlier. The FS-Net is also leveraged in the proposed FDL-SD and FDL-KSD schemes to generate a highly reliable initial candidate so that the search in the SD and KSD can be accelerated in conjunction with candidate/layer ordering and early rejection. The simulation results show that a complexity reduction ratio of more than 90% w.r.t. the conventional/existing algorithms is attained by the proposed schemes with no/marginal performance loss.

Table of contents

List of tables

List of figures

List of symbols

AWGN	additive white Gaussian noise
BER	bit-error rate
BPSK	binary phase-shift keying
BS	base station
CDF	cumulative distribution function
DetNet	detection network
DL	deep learning
DL-TS	deep learning-aided tabu search
DNN	deep neural network
DPP-SD	learning-aided deep path prediction for sphere decoding
ET	early termination
FC-DNN	fully-connected deep neural network
FDL-KSD	fast deep learning-aided K-best sphere decoding
FDL-SD	fast deep learning-aided sphere decoding
FP-SD	Fincke-Pohst sphere decoding
FSD	fixed-complexity sphere decoding
FS-Net	fast-convergence sparsely connected detection network
KSD	K-best sphere decoding
LB	lower bound
LTS	layered tabu search
MIMO	multiple-input multiple-output
ML	maximum-likelihood

MMSE	minimum-mean-square-error
MR-DL-SD	multiple-radius deep learning-aided sphere decoding
MS	mobile station
NG-TS	neighbor-grouped tabu search
OAMP	orthogonal approximate message passing
OFDM	orthogonal frequency-division multiplexing
OSIC	ordered successive interference cancellation
QAM	quadrature amplitude modulation
QPSK	quadrature phase-shift keying
QR-TS	QR-decomposition-aided tabu search
QR-TS-CO	QR-decomposition-aided tabu search with channel ordering
ScNet	sparsely connected detection network
SD	sphere decoding
SE-SD	Schnorr-Euchner sphere decoding
SIC	successive interference cancellation
SNR	signal-to-noise ratio
SR-DL-SD	single-radius deep learning-aided sphere decoding
TS	tabu search
UB	upper bound
ZF	zero-forcing

1. Introduction

In mobile communications, a large multiple-input multiple-output (MIMO) system, where the base station (BS) is equipped with a large number of antennas, has been recently considered as a potential technique for dramatically improving system spectral and power efficiency [1], [2]. However, the promised advantages of large MIMO require significantly increased computational complexity at the receiver compared to the conventional MIMO system [3]. In the conventional MIMO, the sphere decoder (SD) has been widely considered as the most efficient approach to achieve the near-maximum likelihood (ML) performance with reduced complexity compared to the optimal ML detector. However, the lower bound of its complexity exponentially grows with the number of transmit antennas and constellation orders [4], which leads to excessively large complexity in large MIMO systems. The variations of the original SD, such as $K-$best SD (KSD) [5, 6], and fixed-complexity SD (FSD) [7], have been proposed to further reduce or limit the complexity of SD while sacrificing error performance depending on chosen design parameters. However, to guarantee near-ML performance, especially at high signal-to-noise ratios (SNRs), their complexities can be higher than that of the original SD [5], [7], [8]. By contrast, the tabu search (TS) detector can perform very close to the ML bound with far lower complexity compared to SD or FSD in large MIMO systems [9], [10]. However, its complexity is still high due to the iterative search procedure, especially in large MIMO systems, where a large number of neighbors are examined in each iteration. Therefore, low-complexity near-optimal detection is one of the challenges in realizing large MIMO systems, and has thus attracted considerable interests recently [9, 11, 12].

In this thesis, to overcome this challenge, we study low-complexity near-optimal detection schemes for large MIMO systems. By optimizing the tabu search (TS) and sphere decoding (SD) schemes, we propose five novel near-optimal reduced-complexity detection algorithms, namely, the QR-decomposition-aided TS (QR-TS), neighbor-grouped TS (NG-TS), deep learning (DL)-aided TS (DL-TS), fast DL-aided SD (FDL-SD), and fast DL-aided $K-$best SD (KSD) (FDL-KSD), for large MIMO systems. These schemes will be introduced in details in Chapters 2–5 of this thesis.

In this chapter, we first present the background of symbol detection in MIMO systems. Specifically, the system model and the fundamental solutions of the MIMO detection problem, including the optimal ML solution, near-optimal solutions of the SD and TS schemes, sub-optimal zero-forcing (ZF), minimum-mean-square-error (MMSE), and deep neural network (DNN)-based solutions, are presented. Then the literature review are presented with the focus on low-complexity near-optimal detection schemes for large MIMO systems. Finally, the thesis scope and contributions are summarized.

1.1. Background

1.1.1. System model

We consider the uplink of a multi-user MIMO system with N_r receive antennas, where the total number of transmit antennas among all users is N_t. The received signal vector $\tilde{\boldsymbol{y}}$ is given by

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{H}}\,\tilde{\boldsymbol{s}} + \tilde{\boldsymbol{n}}, \tag{1.1}$$

where $\tilde{\boldsymbol{s}} = [\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_{N_t}]^T$ is the vector of transmitted symbols. We assume that $\mathbb{E}\left\{|\tilde{s}_i|^2\right\} = \sigma_t^2$, where σ_t^2 is the average symbol power, and $\tilde{\boldsymbol{n}}$ is a vector of independent and identically distributed (i.i.d.) additive white Gaussian noise (AWGN) samples, $\tilde{n}_i \sim \mathcal{CN}(0, \sigma_n^2)$. Furthermore, $\tilde{\boldsymbol{H}}$ denotes an $N_r \times N_t$ channel matrix consisting of entries $\tilde{h}_{i,j}$, where $\tilde{h}_{i,j}$ represents the complex channel gain between the jth transmit antenna and the ith receive antenna. The transmitted symbols $\tilde{s}_i, i = 1, 2, \ldots, N_t$, are independently drawn from a complex constellation $\tilde{\mathcal{A}}$ of $\tilde{Q}$ points. The set of all possible transmitted vectors forms an N_t-dimensional complex constellation $\tilde{\mathcal{A}}^{N_t}$ consisting of $\tilde{Q}^{N_t}$ vectors, i.e., $\tilde{\boldsymbol{s}} \in \tilde{\mathcal{A}}^{N_t}$.

The complex signal model (1.1) can be converted to an equivalent real signal model

$$\boldsymbol{y} = \boldsymbol{H}\,\boldsymbol{s} + \boldsymbol{n}, \tag{1.2}$$

where

$$\boldsymbol{s} = \begin{bmatrix} \Re(\tilde{\boldsymbol{s}}) \\ \Im(\tilde{\boldsymbol{s}}) \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \Re(\tilde{\boldsymbol{y}}) \\ \Im(\tilde{\boldsymbol{y}}) \end{bmatrix}, \quad \boldsymbol{n} = \begin{bmatrix} \Re(\tilde{\boldsymbol{n}}) \\ \Im(\tilde{\boldsymbol{n}}) \end{bmatrix}, \quad \text{and } \boldsymbol{H} = \begin{bmatrix} \Re\left(\tilde{\boldsymbol{H}}\right) & -\Im\left(\tilde{\boldsymbol{H}}\right) \\ \Im\left(\tilde{\boldsymbol{H}}\right) & \Re\left(\tilde{\boldsymbol{H}}\right) \end{bmatrix},$$

respectively denote the $(N \times 1)$-equivalent real transmitted signal vector, $(M \times 1)$-equivalent real received signal, AWGN noise signal vectors, and $(M \times N)$-equivalent real channel matrix, with $N = 2N_t, M = 2N_r$. Here, $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary parts of a complex vector or matrix, respectively. Then, the set of all possible real-valued transmitted vectors forms an N-dimensional constellation $\mathcal{A}^N$ consisting of Q^N vectors, i.e., $\boldsymbol{s} \in \mathcal{A}^N$. In this work, we use the equivalent real-valued signal model in (1.2) because it can be employed not only for the TS and SD schemes but also for DNNs.

1.1.2. Detection in MIMO systems

1) Optimal ML solution and near-optimal solutions

The ML optimal solution can be written as

$$\hat{\boldsymbol{s}}_{ML} = \arg \min_{\boldsymbol{s} \in \mathcal{A}^N} \phi(\boldsymbol{s}), \tag{1.3}$$

where $\phi(\boldsymbol{s}) = \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|^2$ is the ML metric of $\boldsymbol{s}$. The computational complexity of ML detection in (1.3) is exponential with N [10], which causes extremely high complexity for large MIMO systems, where N is very large.

Similar to the ML detection, the SD attempts to find the optimal lattice point that is closest to $\boldsymbol{y}$, but its search is limited to the points inside a sphere of radius d, i.e.,

$$\hat{\boldsymbol{s}}_{SD} = \arg \min_{\boldsymbol{x} \in \mathcal{S} \subset \mathcal{A}^N} \phi(\boldsymbol{s}), \tag{1.4}$$

where $\mathcal{S}$ is a hypersphere specified by the center $\boldsymbol{y}$ and radius of d.

By contrast, the TS detection scheme employs an iterative neighbor search strategy to find the near-optimal solution. After a sufficiently large searching iterations, the final solution $\hat{\boldsymbol{s}}_{TS}$ is determined as the best candidate visited so far, i.e.,

$$\hat{\boldsymbol{s}}_{TS} = \arg \min_{\boldsymbol{c} \in \mathcal{V}} \{\phi(\boldsymbol{c})\}, \tag{1.5}$$

where $\mathcal{V}$ is the set of all visited candidates over searching iterations. The detailed operations of the SD and TS schemes will be further discussed in Chapters 2–5.

2) Linear solutions

The linear ZF and MMSE solutions are given as

$$\hat{\boldsymbol{s}}_{ZF} = (\boldsymbol{H}^H \boldsymbol{H})^{-1} \boldsymbol{H} \boldsymbol{y},$$
$$\hat{\boldsymbol{s}}_{MMSE} = (\boldsymbol{H}^H \boldsymbol{H} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{H} \boldsymbol{y},$$

respectively. In practical multiuser large MIMO systems, where the number of transmit and receive antennas are comparable, the ZF and MMSE receivers cannot achieve full diversity [13].

3) DNN-based solutions

A deep neural network (DNN) can be modeled and trained to approximate the transmitted signal vector $\boldsymbol{s}$. The solution obtained by a DNN with L layers can be formulated as $\hat{\boldsymbol{s}} = \mathcal{Q}\left(\hat{\boldsymbol{s}}^{[L]}\right)$, where $\mathcal{Q}(\cdot)$ is the element-wise quantization operator that quantizes $\hat{s}_n^{[L]} \in \mathbb{R}$ to $\hat{s}_n \in \mathcal{A}, n = 1, \ldots, N$. Here, $\hat{\boldsymbol{s}}^{[L]}$ is the output vector at the Lth layer, which can be expressed as

$$\hat{\boldsymbol{s}}^{[L]} = f^{[L]}\left(f^{[L-1]}\left(\ldots\left(f^{[1]}\left(\boldsymbol{x}^{[1]}; \boldsymbol{P}^{[1]}\right); \ldots\right); \boldsymbol{P}^{[L-1]}\right); \boldsymbol{P}^{[L]}\right), \tag{1.6}$$

where

$$f^{[l]}\left(\boldsymbol{x}^{[l]}; \boldsymbol{P}^{[l]}\right) = \sigma^{[l]}\left(\boldsymbol{W}^{[l]} \boldsymbol{x}^{[l]} + \boldsymbol{b}^{[l]}\right) \tag{1.7}$$

represents the nonlinear transformation in the lth layer with the input vector $\boldsymbol{x}^{[l]}$, the activation function $\sigma^{[l]}$, and $\boldsymbol{P}^{[l]} = \left\{\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right\}$ consisting of the weighting matrix $\boldsymbol{W}^{[l]}$ and bias vector $\boldsymbol{b}^{[l]}$. We see that (1.6) indicates the serial nonlinear transformations in the DNN that maps the input $\boldsymbol{x}^{[1]}$, including the information contained in $\boldsymbol{y}$ and $\boldsymbol{H}$, to the

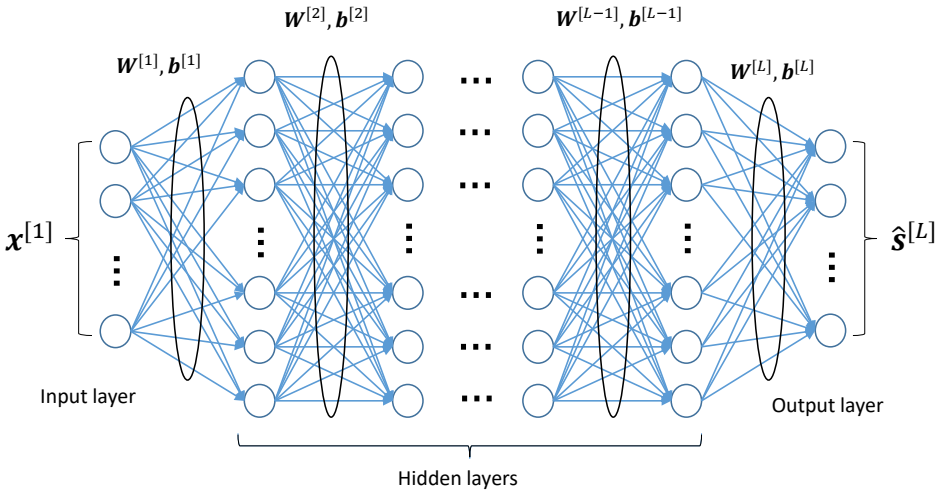


FIG. 1.1. Serial nonlinear transformations in the DNN that maps the input $\boldsymbol{x}^{[1]}$, including the information contained in $\boldsymbol{y}$ and $\boldsymbol{H}$, to the output $\hat{\boldsymbol{s}}^{[L]}$.

output $\hat{\boldsymbol{s}}^{[L]}$, as illustrated in Fig. 1.1.

In large MIMO systems, many hidden layers and neurons are required for the DNN to extract meaningful features and patterns from the large amount of input data to provide high accuracy. Furthermore, in large MIMO systems, $\boldsymbol{x}$ becomes a high-dimensional vector. Therefore, large-size weight matrices and bias vectors, i.e., $\boldsymbol{W}^{[l]}$ and $\boldsymbol{b}^{[l]}$, are required in (1.7). As a result, the computational complexity of the detection network typically becomes very high in large MIMO systems. Therefore, optimizing DNN detection architecture in terms of performance-complexity tradeoff and its applications of DL to symbol detection in large MIMO systems is challenging, which will be further investigated in Chapters 4 and 5.

1.2. Literature review and motivation

In mobile communications, by simultaneously transmitting/receiving as many data streams as possible with large numbers of transmit/receive antennas, the total throughput can be significantly enhanced. Therefore, a large MIMO system is a potential technique to dramatically improve the systems's spectral and energy efficiency. In a practical uplink multiuser massive MIMO system, the numbers of transmit and receive antennas can be comparable [9]. In this scenario, the low-complexity sub-optimal detectors such as linear ZF/MMSE and successive interference cancellation (SIC) receivers cannot achieve full diversity [13]. In contrast, the ML detection performs optimally, but its computational cost increases exponentially with the number of transmit antennas, which is prohibitive in large MIMO systems. Therefore, low-complexity near-optimal detection is an important challenge in optimizing large MIMO systems [9, 11, 12].

Various algorithms for large-MIMO detection have been introduced [13–23]. In [13], a high-performance, low-complexity detector called likelihood ascent search (LAS) and its extension, multistage LAS [14] are proposed for large-MIMO systems with binary phase-shift keying (BPSK) modulation. The work in [17] considers the application of belief propagation to achieve near-optimal signal detection with low complexity for 64 $\times$ 64 MIMO with BPSK and 4-quadrature amplitude modulation (QAM). In [18], low-complexity probabilistic-data-association-based decoding algorithms are proposed for non-orthogonal space-time block-coded large MIMO systems; their performance approached the single-input single-output AWGN performance with 4-QAM but the performance gap becomes larger for higher-order modulations such as 16-QAM. In [19], an algorithm based on Monte Carlo sampling, known as the mixed Gibbs sampling algorithm, is proposed

to alleviate the stalling problem in [20]. However, MGS provides poor performance in higher-order QAM modulations; this problem is alleviated in [21]. Most previous works focus on low-order modulation (e.g., BPSK, 4-QAM) while higher-order QAM can be used to convey information at higher rates. To address this, channel-hardening-based [22] and soft-heuristic [23] algorithms are proposed in the literature for large MIMO systems such as 64×64 MIMO with QPSK and 16-QAM.

Additionally, the TS detector, which is a local neighbor search algorithm, was also introduced as a complexity-efficient scheme for symbol detection in large MIMO systems. It has been shown that the TS detection algorithm can perform very close to the ML bound with far lower complexity compared to SD or FSD in large MIMO systems [9], [10]. In [24], the authors present an approach based on reactive tabu search for near-ML decoding of non-orthogonal 64×64 STBCs with 4-QAM. However, its performance is far from optimum for higher-order QAMs such as 16- and 64-QAM [25]. The work in [10] proposes an algorithm called layered TS (LTS). This algorithm improves the conventional TS in terms of higher-order QAM performance in large MIMO systems. However, to achieve the bit-error rate (BER) of 10^{-2} in 32×32 and 64×64 MIMO systems with 16-QAM, higher complexities are required compared to conventional TS. The random-restart reactive TS (R3TS) algorithm, which runs multiple reactive TS (RTS) and chooses the best among the resulting solution vectors, is presented in [26]. It achieves improved BER performance at the expense of increased complexity. The complexity of R3TS is generally higher than that of RTS to achieve 10^{-2} BER, especially for large systems and high-order QAMs such as 64×64 MIMO with 64-QAM. Several works have been conducted to further improve TS in terms of complexity by reducing the number of examined neighbors or using a stopping criterion [27], [28]. However, it comes at the cost of performance loss; for example, to achieve BER $= 10^{-3}$ for 4×4 MIMO with 16-QAM modulation, the TS algorithm with a

stopping criterion has a 3-dB SNR loss compared to the original TS [27].

On the other hand, the application of deep learning (DL) to symbol detection in MIMO systems has recently gained much attention [29–34]. In [29], three detection algorithms based on deep neural networks (DNNs) are proposed for molecular communication systems, which are shown to perform much better than the prior simple detectors. In contrast, the application of DL to symbol detection in orthogonal frequency-division multiplexing (OFDM) systems are considered in [30]. Specifically, Ye *et al.* in [30] show that the detection scheme based on DL can address channel distortion and detect the transmitted symbols with performance comparable to that of the MMSE receiver. In [34], the DL-based SD scheme is proposed. In particular, the DL-based SD with the radius of the decoding hypersphere learned by a DNN achieves significant complexity reduction with respect to (w.r.t.) the conventional SD with a marginal performance loss. In particular, the works of [31] and [32] focus on the design of DNNs for symbol detection in large MIMO systems. Specifically, Samuel *et al.* in [31] and [32] first investigate the fully connected DNN (FC-DNN) architecture for symbol detection and show that although it performs well for fixed channels, its BER performance is very poor for varying channels. To overcome this problem, a DNN that works for both fixed and varying channels, called the detection network (DetNet), is introduced [31, 32]. However, the DetNet requires high computational complexity because of its complicated network architecture, motivating the proposal of the sparsely connected network (ScNet) in [33] to improve performance and reduce complexity.

1.3. Thesis scope and contribution

In this thesis, we study low-complexity near-optimal detection schemes for practical multiuser massive MIMO systems. We focus on the TS and SD schemes, which are considered the efficient symbol detection algorithms for large MIMO systems because it performs especially close to the ML bound with reduced complexity. By optimizing the conventional TS and SD schemes, we propose five novel near-optimal reduced-complexity detection schemes, namely, the QR-TS [35], NG-TS [36], deep DL-TS [37], FDL-SD, and FDL-KSD, for large MIMO systems. The major contributions are presented in Chapters 2–5, which can be summarized as follows:

In Chapter 2, the QR-TS scheme [35] is proposed, which allows finding the best neighbor with reduced complexity by exploiting efficient computations and early rejection. The analytical expressions for the average computational complexities to find the best neighbor in the conventional and QR-TS algorithms are derived. Furthermore, we exploit ordering schemes, namely, transmit-ordering (Tx-ordering) and receive-ordering (Rx-ordering), to further reduce the complexity of the QR-TS algorithm. We analytically and numerically justify that the QR-TS with ordering schemes are powerful in larger-sized MIMO systems.

In Chapter 3, a complexity-efficient strategy for neighbor examination is introduced in the proposed NG-TS algorithm [36]. By expanding the ML cost function, we show that the best one can be determined using a simplified cost function. Furthermore, we develop the groupwise neighbor-examination scheme for the TS algorithms, which allows the best neighbor in each iteration to be found with much lower complexity than that of the sequential neighbor-examination approach used in prior TS schemes. In addition, based on the complexity analysis of the NG-TS algorithm, we propose a channel ordering

scheme for further complexity reduction.

In Chapter 4, the TS detection scheme is optimized by leveraging the power of DL. First, we develop the FS-Net, which achieves improved performance and reduced complexity w.r.t. the prior networks, namely, DetNet and ScNet. As a result, the FS-Net-based solution is taken as the initial solution of the TS algorithm. Secondly, we improve the iterative searching phase of the TS algorithm and propose using an adaptive cutoff factor based on the FS-Net's output. As a result, when the initial solution is likely to be accurate, a small number of searching iterations is taken, which leads to a reduction in the overall complexity of the TS algorithm.

In chapter 5, we propose the FDL-SD and FDL-KSD algorithms, which can overcome the limitations of the existing DL-aided SD schemes by a novel application of DL to the SD. Specifically, we use a DNN to generate a highly reliable initial candidate for the search in the SD, rather than generating the radius as done in the existing DL-aided SD schemes in [34, 38, 39]. Furthermore, the output of the DNN facilitates a candidate/layer ordering scheme and an early rejection scheme for significant complexity reduction. The proposed ideas are applicable to both the sequential SD and KSD and exhibit a considerable improvement in the performance-complexity tradeoff w.r.t. the conventional SD and KSD.

2. QR-decomposition-aided Tabu Search Detection for Large MIMO Systems

2.1. Motivation and contribution

TS detection is considered one of the efficient symbol detection algorithms for large MIMO systems because it performs especially close to the ML bound with a significantly lower complexity compared with SD and FSD in large MIMO systems, as shown in [9] and [10]. In this work, we optimize the conventional TS algorithm in terms of computational complexity for symbol detection in large MIMO with both lower-order (e.g., QPSK) and higher-order (e.g., 16- and 64-QAM) modulations. In each searching iteration of TS detection, the algorithm needs to find the best neighboring vector of the current candidate, which is computationally expensive, especially in large MIMO. This motivates us to develop an algorithm to efficiently find the best neighbor, which can significantly reduce the overall complexity. The main contributions include:

- We propose the QR-TS algorithm based on the QR decomposition of the channel matrix, which allows finding the best neighbor with reduced complexity by exploiting efficient computations and early rejection while achieving exactly the same BER performance with conventional TS.

- The analytical expressions for the average computational complexities to find the best neighbor in the conventional and QR-TS algorithms are derived. This allows us to analytically compare the proposed algorithm to the conventional one in terms of computational complexity and further optimize the QR-TS algorithm.

- We exploit ordering schemes, namely, transmit-ordering (Tx-ordering) and receive-ordering (Rx-ordering), to further reduce the complexity of the QR-TS algorithm. We analytically and numerically justify that the ordering schemes are powerful in larger-sized MIMO systems. Simulation results show that QR-TS reduces the computational complexity of the conventional TS algorithm by approximately half. Moreover, if the ordering schemes are incorporated, they require only approximately one-fourth the complexity of the conventional TS. It is also shown that the proposed ordered TS schemes can achieve complexity reduction gain in both lower- and higher-order modulations.

The rest of this chapter is organized as follows: Section 2.2 presents the conventional TS algorithm. Section 2.3 describes the proposed QR-TS scheme and analyzes its complexity reduction. Moreover, ordering schemes that further reduce the complexity of the QR-TS algorithm are also introduced in Section 2.3. Section 2.4 discusses the performance and complexity results of the proposed schemes for various MIMO configurations. Finally, conclusions are presented in Section 2.5.

2.2. Conventional TS algorithm

In this subsection, the main concepts of the conventional TS detection algorithm are briefly explained. We focus on the analysis of the computational complexity of this algorithm, which is useful for comparison with the proposed algorithm in Section 2.3.3. For a detailed description of the TS algorithm, please refer to [10, 24].

The conventional TS algorithm searches candidates and their neighbors over $\mathcal{I}$ iterations. It uses a tabu list $\mathcal{L}$ of length $P\,(P \leq \mathcal{I})$ to record the visited points to avoid cycling. The output of the TS algorithm is the best solution vector $\hat{\boldsymbol{s}}_{TS}$ found until $\mathcal{I}$ searching iterations.

2.2.1. Initialization phase

The algorithm starts with an initial candidate vector, which is assumed to be the ZF solution $\boldsymbol{x}_{ZF} = \ulcorner \boldsymbol{H}^{\dagger}\boldsymbol{y} \urcorner$, where $\boldsymbol{H}^{\dagger}$ is the pseudoinverse of $\boldsymbol{H}$ and $\ulcorner \cdot \urcorner$ is a quantized vector. With the assumption of $N_t = N_r$ and full rank of $\boldsymbol{H}$, we have $\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{-1}$ and $\boldsymbol{x}_{ZF} = \ulcorner \boldsymbol{H}^{-1}\boldsymbol{y} \urcorner$. Assuming that the well-known Gaussian elimination method is used, solving for $\boldsymbol{x}_{ZF}$ requires $\frac{1}{3}N^3 + \frac{1}{2}N^2 - \frac{5}{6}N$ multiplications and $\frac{1}{3}N^3 + \frac{1}{2}N^2 - \frac{5}{6}N$ additions [40]. Furthermore, the metric of the initial candidate is computed as $\phi\left(\boldsymbol{c}_{\{1\}}\right) = \left\|\boldsymbol{y} - \boldsymbol{H}\,\boldsymbol{c}_{\{1\}}\right\|^2$, which requires $N^2 + N$ multiplications and $N^2 + N - 1$ additions. Consequently, the total complexity required for the initialization phase becomes $\frac{1}{3}N^3 + \frac{3}{2}N^2 + \frac{1}{6}N$ multiplications and $\frac{1}{3}N^3 + \frac{3}{2}N^2 + \frac{1}{6}N - 1$ additions.

TABLE 2.1. Maximum number of neighbors of a candidate $\boldsymbol{c}$, with α is the number of elements of $\boldsymbol{c}$ having the highest absolute amplitude level, $0 \leq \alpha \leq N$

Modulation scheme	BPSK	QPSK	16-QAM	64-QAM
Maximum number of neighbors	$N-1$	$N-1$	$2N-\alpha-1$ $\alpha=3$	$2N-\alpha-1$ $\alpha=7$

2.2.2. Searching phase

In each searching iteration, the TS algorithm examines the neighboring vectors of the current candidate and subsequently moves to the best neighboring vector. In this work, neighbor $\boldsymbol{x}$ of a given candidate $\boldsymbol{c}$ is defined as a non-tabu vector inside alphabet $\mathcal{A}^N$ with the smallest distance to $\boldsymbol{c}$. Therefore, the neighbor set of $\boldsymbol{c}$ is given by

$$\mathcal{N}(\boldsymbol{c}) = \left\{ \boldsymbol{x} \in \mathcal{A}^N \backslash \mathcal{L}, |\boldsymbol{x} - \boldsymbol{c}| = \theta_{min} \right\}, \tag{2.1}$$

where $\mathcal{A}^N \backslash \mathcal{L}$ denotes the alphabet $\mathcal{A}^N$ excluding the tabu vectors kept in $\mathcal{L}$, and θ_{min} is the minimum distance between two constellation points in a plane. The maximum numbers of neighbors in $\mathcal{N}(\boldsymbol{c})$ for different modulation schemes are presented in Table 2.1. It is noteworthy that for BPSK and QPSK modulations, each candidate vector $\boldsymbol{c}$ has a maximum of N neighboring vectors. However, in higher-order modulation schemes, such as 16- and 64-QAM, the number of neighbors of $\boldsymbol{c}$ depends on its elements. The elements with the highest absolute amplitude level have only a single neighboring symbol each, whereas each of the other elements has two neighboring symbols. Therefore, in higher-order QAM schemes, the maximum number of neighbors of $\boldsymbol{c}$ is given by $2N - \alpha$, where $\alpha \in [0, N]$ is the number of elements of $\boldsymbol{c}$ having the highest absolute amplitude level. The subtractions to one in Table 2.1 imply the exclusion of the neighbor that is maintained in the tabu list to avoid cycling.

In $\mathcal{N}(\boldsymbol{c})$, the best neighbor $\boldsymbol{x}^\star$ is the one with the smallest metric, i.e.,

$$\boldsymbol{x}^\star = \arg \min_{\boldsymbol{x} \in \mathcal{N}(\boldsymbol{c})} \phi(\boldsymbol{x}).$$

In every iteration, the cost metric of $\boldsymbol{x}^\star$ is compared to that of $\hat{\boldsymbol{s}}_{TS}$. If a new smaller metric is found, i.e., $\phi(\boldsymbol{x}^\star) < \phi(\hat{\boldsymbol{s}}_{TS})$, the algorithm updates the best solution $\hat{\boldsymbol{s}}_{TS}$ to $\boldsymbol{x}^\star$. It is worth noting that in the TS algorithm, to escape from local minima, the move from $\boldsymbol{c}$ to $\boldsymbol{x}^\star$ is made in every iteration even when $\boldsymbol{x}^\star$ is worse than $\boldsymbol{c}$ in terms of the ML cost, i.e., $\phi(\boldsymbol{x}^\star) > \phi(\boldsymbol{c})$. After each move, the current candidate and tabu list are updated as $\boldsymbol{c}_{\{i\}} = \boldsymbol{x}^\star_{\{i-1\}}$ and $\mathcal{L}_{\{i\}} = \left\{\mathcal{L}_{\{i-1\}}, \boldsymbol{x}^\star_{\{i-1\}}\right\}$, where the subscript $\{i\}$ indicates the ith iteration. If the tabu list is full, its first element will be released to make space for another vector. This process is terminated after $\mathcal{I}$ iterations or if a termination condition is satisfied, and the final solution is determined as the best solution vector found.

The metric of a neighbor $\boldsymbol{x}$ can be expressed as

$$\phi(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2 = \|\boldsymbol{u} + \boldsymbol{H}\Delta\boldsymbol{x}\|^2 = \|\boldsymbol{u} + \boldsymbol{h}_d\delta_d\|^2, \tag{2.2}$$

where $\boldsymbol{u} = \boldsymbol{y} - \boldsymbol{H}\boldsymbol{c}$, $\Delta\boldsymbol{x} = \boldsymbol{c} - \boldsymbol{x} = [0, \ldots, 0, \delta_d, 0, \ldots, 0]^T$ has only one nonzero element $\delta_d = c_d - x_d$, and $\boldsymbol{h}_d$ is the dth column of $\boldsymbol{H}$. In this study, we refer to d as the "difference position" of a neighbor. For example, if the current candidate is $\boldsymbol{c} = [1, -3, 1, 3]^T$, then $d = 3$ is the difference position for $\boldsymbol{x} = [1, -3, -1, 3]^T$ because $\boldsymbol{c}$ and $\boldsymbol{x}$ are only different w.r.t. the third element. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_L\}$ be the set of difference positions of L neighbors in $\mathcal{N}(\boldsymbol{c})$. Since $\Delta\boldsymbol{x}$ is a single-nonzero vector, we obtain the last equation in (2.2). Let $\bar{L}$ be the average number of neighbors in an iteration. From (2.2), the average complexity to find the best neighbor in an iteration includes $2\bar{L}N$ multiplications and $\bar{L}(2N-1)$ additions. Furthermore, $\bar{L}$ comparisons on average are required to determine

and update the best neighbor in each searching iteration of the TS algorithm. It is observed that the computational load of TS comes primarily from the computations of the neighbors' metrics to find the best neighbor in each iteration. This motivates us to develop a novel TS algorithm that allows determining the best neighbor with low complexity to reduce the overall computational load of the TS algorithm.

2.3. Proposed QR-TS algorithm

In this section, we introduce the proposed QR-TS algorithm that aims to optimize the conventional TS in terms of computational complexity. The complexity analysis is also given in this section.

2.3.1. Basic ideas

The metric in (2.2) can be rewritten as

$$\phi(\boldsymbol{x}) = \sum_{n=1}^{N} |u_n + h_{n,d}\delta_d|^2 . \tag{2.3}$$

It implies that to compute $\phi(\boldsymbol{x})$, the algorithm needs to consider N terms, i.e., $|u_n + h_{n,d}\delta_d|^2$, $n = 1, \ldots, N$. However, in large MIMO systems, N is large, and hence, the computation of $\phi(\boldsymbol{x})$ becomes computationally expensive. The QR-TS algorithm allows the computation of $\phi(\boldsymbol{x})$ with considerably lower complexity as a benefit of the QR decomposition of the channel matrix $\boldsymbol{H}$. Furthermore, it also computes the neighbors' metrics more efficiently by accumulating only a subset of N terms.

By using the QR decomposition, we have $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q}$ is a unitary matrix, and $\boldsymbol{R}$ is an upper triangular matrix. Subsequently, the cost metric of $\boldsymbol{x}$ can be rewritten as

$$\begin{aligned}\phi(\boldsymbol{x}) &= \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2 = \left\|\boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}\boldsymbol{x}\right\|^2 \\ &= \left\|\boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}\right\|^2 = \left\|\boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\left(\boldsymbol{c} - \Delta\boldsymbol{x}\right)\right\|^2. \end{aligned} \tag{2.4}$$

By defining $\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}$, (2.4) can be rewritten as

$$\phi(\boldsymbol{x}) = \|\boldsymbol{z} + \boldsymbol{R}\Delta\boldsymbol{x}\|^2 = \|\boldsymbol{z} + \boldsymbol{r}_d\delta_d\|^2 \tag{2.5}$$

$$= \underbrace{\sum_{n=1}^{d} |z_n + r_{n,d}\delta_d|^2}_{\triangleq\phi_A} + \underbrace{\sum_{n=d+1}^{N} |z_n|^2}_{\triangleq\phi_B}, \tag{2.6}$$

where $\boldsymbol{r}_d$ is the dth column of $\boldsymbol{R}$, and z_n is the nth element of $\boldsymbol{z}$. Equation (2.6) motivates the development of two main ideas to reduce the computational complexity of QR-TS, which are efficient metric computation and early rejection.

1) Efficient metric computation

In $\boldsymbol{r}_d$, only the first d elements are non-zeros, which makes the sum ϕ_B in (2.6) independent of $\boldsymbol{x}$. Therefore, ϕ_B in (2.6) can be computed for all neighbors with minimal complexity with the following techniques:

• *Within one searching iteration,* $\boldsymbol{z}$ is constant for every neighbor since $\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}$, whereas $\boldsymbol{Q}$, $\boldsymbol{R}$, and $\boldsymbol{y}$ are fixed for $\mathcal{I}$ iterations, and $\boldsymbol{c}$ is invariant with neighbors. Therefore, the algorithm needs to compute $\sum_{n=k+1}^{N} |z_n|^2, k = 1, 2, \ldots, N-1$ only once and store them

in an array $\boldsymbol{e}$ to reuse it for every neighbor. Here, we define

$$\boldsymbol{e} = \left[\sum_{n=2}^{N} |z_n|^2, \sum_{n=3}^{N} |z_n|^2, \ldots, |z_N|^2, 0\right]. \tag{2.7}$$

Obviously, the kth element of $\boldsymbol{e}$, e_k, can be recursively computed with low complexity by $e_k = e_{k+1} + |z_{k+1}|^2$, $k = 1, 2, \ldots, N-1$, and $e_N = 0$. After computing $\boldsymbol{e}$, we use it to obtain the neighbors' metrics:

$$\phi(\boldsymbol{x}) = e_d + \sum_{n=1}^{d} |z_n + r_{n,d}\delta_d|^2. \tag{2.8}$$

This idea is further illustrated in Table 2.2. It is shown that the sums ϕ_B of different neighbors have many common terms, which allows QR-TS to save a substantial number of operations compared to the conventional algorithm.

- *Over two successive searching iterations,* only a few first elements of $\boldsymbol{z}$ are updated. Note that between two successive iterations, i and $i+1$, $\boldsymbol{c}_{\{i\}}$ and $\boldsymbol{c}_{\{i+1\}}$ are neighbors of each other. In particular, $\boldsymbol{c}_{\{i+1\}}$ is the best neighbor of $\boldsymbol{c}_{\{i\}}$, i.e., $\boldsymbol{x}^{\star}_{\{i\}} = \boldsymbol{c}_{\{i+1\}}$. Let $d^{\star}$ be the difference position of $\boldsymbol{x}^{\star}_{\{i\}}$. Then, $\boldsymbol{c}_{\{i\}}$ and $\boldsymbol{c}_{\{i+1\}}$ are different only in the $d^{\star}$th element. Therefore, we have

$$\begin{aligned}
\boldsymbol{z}_{\{i+1\}} - \boldsymbol{z}_{\{i\}} &= \left(\boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}_{\{i+1\}}\right) - \left(\boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}_{\{i\}}\right) \\
&= \boldsymbol{R}\left(\boldsymbol{c}_{\{i\}} - \boldsymbol{c}_{\{i+1\}}\right) = \boldsymbol{R}\Delta\boldsymbol{x}_{\{i\}} = \boldsymbol{r}_{d^{\star}}\delta_{d^{\star}} \\
&= [r_{1,d^{\star}}\delta_{d^{\star}}, \ldots, r_{d^{\star},d^{\star}}\delta_{d^{\star}}, 0, \ldots, 0]^T,
\end{aligned}$$

which implies that over two successive iterations, only the first $d^{\star}$ elements of $\boldsymbol{z}$ should be updated. Hence, in Table 2.2, the algorithm only needs to update the first $d^{\star}-1$ elements of $\boldsymbol{e}$, i.e., $e_1, e_2, \ldots, e_{d^{\star}-1}$, and the complexity for the others can be saved. Because it is

TABLE 2.2. Neighbors' metric

d	Metric of the neighbor corresponding to d
1	$\underbrace{\lvert z_N\rvert^2 + \ldots\ldots\ldots + \lvert z_3\rvert^2 + \lvert z_2\rvert^2}_{\phi_B = e_1} + \lvert z_1 + r_{1,1}\delta_1\rvert^2$
2	$\underbrace{\lvert z_N\rvert^2 + \ldots\ldots\ldots + \lvert z_3\rvert^2}_{\phi_B = e_2} + \sum_{n=1}^{2} \lvert z_n + r_{n,2}\delta_2\rvert^2$
...	...
d	$\underbrace{\lvert z_N\rvert^2 + \ldots + \lvert z_{d+1}\rvert^2}_{\phi_B = e_d} + \sum_{n=1}^{d} \lvert z_n + r_{n,d}\delta_d\rvert^2$
...	...
$N-1$	$\underbrace{\lvert z_N\rvert^2}_{\phi_B = e_{N-1}} + \sum_{n=1}^{N-1} \lvert z_n + r_{n,N-1}\delta_{N-1}\rvert^2$
N	$\underbrace{0}_{e_N} + \sum_{n=1}^{N} \lvert z_n + r_{n,N}\delta_N\rvert^2$

relatively cheap to compute ϕ_B for the neighbors, it is first computed in the process of calculating the cumulative metrics of neighbors. These advantages of QR-TS can significantly reduce the complexity of the TS algorithm, which will be further discussed in Section 2.3.3.

In prior works regarding MIMO detection, the QR decomposition is used to simplify the metric function [41, 42], which results in overall complexity reduction. However, we note that in the proposed QR-TS algorithm, the QR decomposition is incorporated into not only the simplified cost metric but also the efficient metric computation, as described in this subsection, which can potentially result in larger complexity reduction.

2) Early rejection

It is shown in (2.8) that to calculate the metric of the neighbor $\boldsymbol{x}$, d terms, i.e., $|z_1 + r_{1,d}\delta_d|^2$, ..., $|z_d + r_{d,d}\delta_d|^2$, need to be computed. It requires lower complexity than the conventional TS algorithm, where N terms are calculated as in (2.3). However, it is still inefficient and requires high complexity because of the large number of neighbors in large MIMO systems. In fact, the neighbors' metrics are distributed in a wide range, and

a subset of neighbors have much larger metrics compared to the others. If these neighbors can be rejected early, the complexity of the TS algorithm can be reduced without causing any performance loss. Therefore, instead of considering the metric in (2.8), the QR-TS scheme computes the neighbors' metrics recursively as follows:

$$\begin{aligned}\phi_\eta(\boldsymbol{x}) &= \phi_B + \sum_{n=1}^{\eta} |z_n + r_{n,d}\delta_d|^2 \\ &= \phi_{\eta-1}(\boldsymbol{x}) + |z_\eta + r_{\eta,d}\delta_d|^2, \eta \leq d,\end{aligned} \tag{2.9}$$

where $\phi_\eta(\boldsymbol{x})$ is the cumulative metric of $\boldsymbol{x}$ at the layer η, and $\phi_0(\boldsymbol{x}) = \phi_B = e_d$. Whether to reject or to keep a neighbor, after each layer, the cumulative metric $\phi_\eta(\boldsymbol{x})$ is compared to the threshold γ. As soon as $\phi_\eta(\boldsymbol{x})$ exceeds γ, the algorithm stops examining this neighbor. For $\eta < d$, we have $\phi_\eta(\boldsymbol{x}) < \phi(\boldsymbol{x})$, and $\phi_\eta(\boldsymbol{x})$ is referred to as an "incomplete metric." In contrast, the term "complete metric" indicates $\phi_\eta(\boldsymbol{x})$ for $\eta = d$. It is obvious that an incomplete metric requires lower complexity than a complete metric. To guarantee that the exact best neighbor is found, the threshold γ is set to infinity initially. Subsequently, if a smaller-than-γ complete metric is found, both γ and $\boldsymbol{x}^\star$ are updated. Through this process, after all neighbors are examined, γ finally becomes the smallest complete metric among the neighbors' complete metrics, and the best neighbor is the one with the metric γ, i.e., $\phi(\boldsymbol{x}^\star) = \gamma$. Consequently, the early rejection techniques allows QR-TS to find the best neighbor without computing all L complete metrics, which results in an overall complexity reduction.

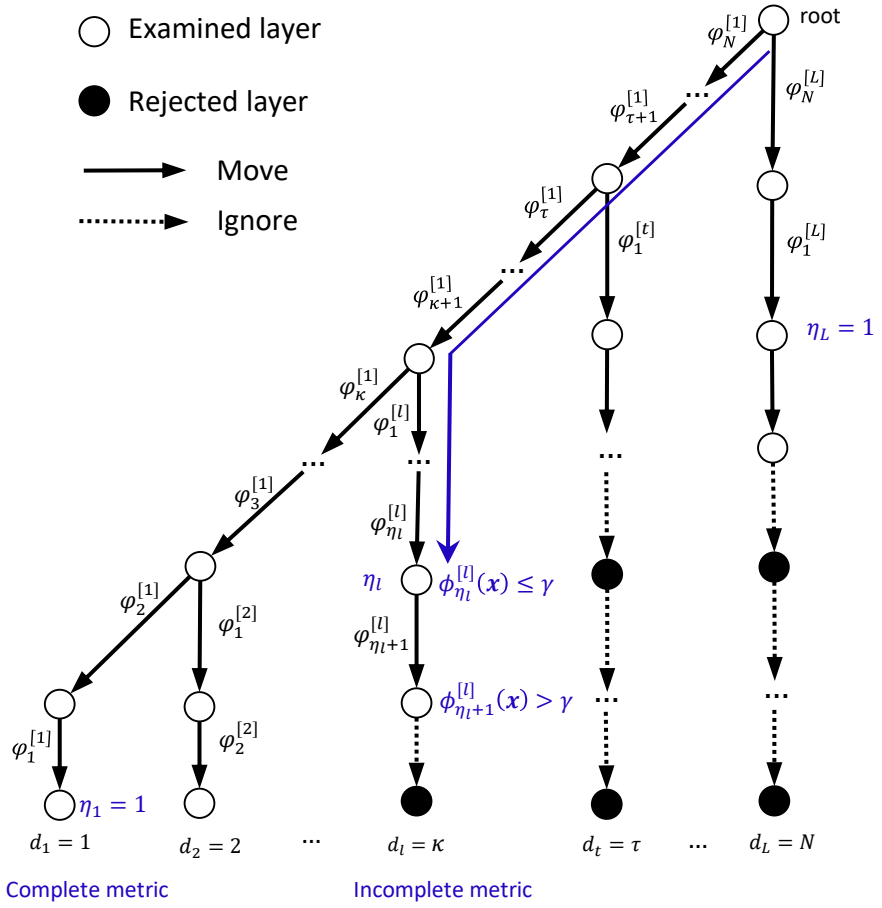


FIG. 2.1. Low-complexity computation and early rejection of QR-TS.

The application of efficient metric computations and early rejection is further illustrated in Fig. 2.1. In this tree diagram, $\varphi_n^{[l]}$ given by

$$\varphi_n^{[l]} = \begin{cases} |z_n|^2, & n > d \\ |z_n + r_{n,d}\delta_d|^2, & n \le d \end{cases}$$

Algorithm 1 Find the best neighbor in the QR-TS algorithm

Input: $\boldsymbol{c}, \boldsymbol{z}, \boldsymbol{R}, \mathcal{N}(\boldsymbol{c}), \mathcal{D}$, and $d^\star$.
Output: The best neighbor $\boldsymbol{x}^\star$.

$e_N = 0$
for $k = d^\star - 1 \to 1$ **do**
  $e_k = e_{k+1} + |z_{k+1}|^2$
end for
$\gamma = \infty, \boldsymbol{x}^\star = \boldsymbol{x}_{[1]}$
for $l = 1 \to L$ **do**
  $\boldsymbol{x} = \boldsymbol{x}_{[l]}, d = d_l$
  $\eta = 1, \phi_\eta(\boldsymbol{x}) = e_d$
  for $\eta = 1 \to d$ **do**
    if $\phi_\eta(\boldsymbol{x}) \leq \gamma$ **then**
      $\phi_\eta(\boldsymbol{x}) = \phi_\eta(\boldsymbol{x}) + |z_\eta + r_{\eta,d}\delta_d|^2$
    else
      break
    end if
  end for
  if $\phi_\eta(\boldsymbol{x}) < \gamma$ **then**
    $\gamma = \phi_\eta(\boldsymbol{x})$
    $\boldsymbol{x}^\star = \boldsymbol{x}$
  end if
end for

represents the metric of layer n of the lth neighbor with $n = 1, 2, \ldots, N$ and $l = 1, 2, \ldots, L$. Therefore, the incomplete metric of the lth neighbor, i.e., $\phi_{\eta_l}^{[l]}(\boldsymbol{x})$, is given by

$$\phi_{\eta_l}^{[l]}(\boldsymbol{x}) = \sum_{n=d+1}^{N} \varphi_n^{[l]} + \sum_{n=1}^{\eta_l} \varphi_n^{[l]} = e_l + \sum_{n=1}^{\eta_l} \varphi_n^{[l]}. \tag{2.10}$$

In Fig. 2.1, the diagonal paths from the root represent the steps to find the vector $\boldsymbol{e}$ in (2.7) with low complexity. By contrast, the vertical paths show the process to compute the second sum in (2.10) to find the complete/incomplete metrics of the neighbors. For example, the early rejection process of the lth neighbor with difference position κ is illustrated by the κth vertical path in the figure. In this example, when $\phi_{\eta_l+1}^{[l]}(\boldsymbol{x}) > \gamma$, the QR-TS algorithm rejects this neighbor and moves on to the next neighbor.

The operation of the proposed QR-TS to find the best neighbor is summarized in Algorithm 1. Steps 1–4 are to find the sum ϕ_B for every neighbor, which is then stored in array $\boldsymbol{e}$. It is noted that all elements of $\boldsymbol{e}$ need to be computed in the first iteration. However, from the second iteration, only the first $d^\star - 1$ elements of $\boldsymbol{e}$ are updated, where $d^\star$ is the difference position between the current candidate $\boldsymbol{c}_{\{i\}}$ and the previous candidate $\boldsymbol{c}_{\{i-1\}}$. In steps 5 and 7, $\boldsymbol{x}_{[l]}$ is the lth element in the neighbor set. In steps 8–15, $\boldsymbol{e}$ is used to find the incomplete and complete metrics of neighbors. Steps 16–19 update the threshold γ and the best neighbor as soon as a smaller-than-γ complete metric is found.

2.3.2. QR-TS with ordering schemes

1) Receive ordering (Rx-ordering)

In the computation of ϕ_A in (2.6), if the elements of received signals are arranged in the decreasing order of layer metrics $|z_n + r_{n,d}\delta_d|^2, n = 1, 2, \ldots, d$, the cumulative metric $\phi(\boldsymbol{x})$ can exceed the threshold γ earlier. As a result, the computational complexity in (2.6) can be reduced. Motivated by this fact, in the Rx-ordering scheme, we sort the elements of $\boldsymbol{z} + \boldsymbol{R}\Delta\boldsymbol{x}$ by the expectations of layer metrics, i.e., $\mathbb{E}\left\{|z_n + r_{n,d}\delta_d|^2\right\}$, which is given in the following lemma.

Lemma 2.1. *The average layer metric can be expressed as*

$$\mathbb{E}\left\{|z_n + r_{n,d}\delta_d|^2\right\} \approx \sigma_v^2 + |r_{n,d}|^2 \delta^2, \tag{2.11}$$

where $\delta^2 = |\delta_d|^2$.

Proof. See Appendix A. □

As a result, within an iteration, the expected metric of the nth layer of a neighbor only depends on $|r_{n,d}|^2$. In Rx-ordering, we aim to accelerate the increase of the cumulative metric. Therefore, the layer with the largest metric should be considered first. In other words, for the computation of ϕ_A in Rx-ordered QR-TS, the terms corresponding to the larger $|r_{n,d}|^2$ are computed and accumulated earlier. It is worth noting that Rx-ordering only changes the computation order in the summation expressed in (2.9) and does not cause any changes in $\boldsymbol{R}$ and $\boldsymbol{Q}$.

2) Transmit ordering (Tx-ordering)

If we order the channel columns such that the neighbors with smaller cost metrics are examined earlier, the threshold γ will decrease more quickly, which certainly reduces the complexity of QR-TS by increasing the chances of early rejection. This motivates the development of an ordering scheme at the transmit side that is "Tx-ordering." Intuitively, the cost metrics of all neighbors depend on the channel matrix. However, in the QR-TS algorithm, within one searching iteration, the cost metrics of different neighbors depend on different channel columns $\boldsymbol{r}_d$, as shown in (2.5). Therefore, an ordering scheme based on channel columns can result in complexity reduction. For the development of Tx-ordering, we investigate the average metric of a neighbor, which is given in the following lemma.

Lemma 2.2. *The average metric of a neighbor is given as*

$$\mathbb{E}\left\{\phi\left(\boldsymbol{x}\right)\right\} = N\sigma_v^2 + \delta^2 \left\|\boldsymbol{h}_d\right\|^2. \tag{2.12}$$

Proof. See Appendix B. □

In (2.12), N, σ_v^2, and δ^2 are constant. Therefore, the average metric of a neighbor depends only on the norm of the corresponding channel column, i.e., $\|\boldsymbol{h}_d\|$, which implies that the

neighbors can be ordered optimally based on the norms of the channel columns.

In the QR-TS algorithm, in contrast to the conventional TS, the complexity in metrics computation is different among the neighbors. Specifically, it is shown from (2.6) that the computation of a neighbor metric with smaller d requires lower complexity. Therefore, the neighbors with the smaller value d should be considered earlier. If we denote $\underline{\mathcal{D}} = \left\{d_{[1]}, d_{[2]}, \ldots, d_{[L]}\right\}$ as the increasing order of elements of $\mathcal{D}$, the neighbors should be examined in the order $\underline{\mathcal{N}}(\boldsymbol{c}) = \left\{\boldsymbol{x}_{[1]}, \boldsymbol{x}_{[2]}, \ldots, \boldsymbol{x}_{[L]}\right\}$, where $\boldsymbol{x}_{[l]}$ corresponds to $d_{[l]}$. Hence, (2.12) can be rewritten as

$$\mathbb{E}\left\{\phi\left(\boldsymbol{x}_{[l]}\right)\right\} = N\sigma_v^2 + \delta^2 \left\|\boldsymbol{h}_{d_{[l]}}\right\|^2. \tag{2.13}$$

However, in Tx-ordering, to accelerate the decreasing of the threshold γ, the neighbor with a smaller average metric should be examined earlier. Therefore, the neighbors in $\underline{\mathcal{N}}(\boldsymbol{c})$ should be sorted such that $\mathbb{E}\left\{\phi\left(\boldsymbol{x}_{[1]}\right)\right\} \leq \mathbb{E}\left\{\phi\left(\boldsymbol{x}_{[2]}\right)\right\} \leq \ldots \leq \mathbb{E}\left\{\phi\left(\boldsymbol{x}_{[L]}\right)\right\}$, which is guaranteed by the condition $\|\boldsymbol{h}_1\|^2 \leq \|\boldsymbol{h}_2\|^2 \leq \ldots \leq \|\boldsymbol{h}_N\|^2$, as indicated by (2.13). In conclusion, in Tx-ordering, the channel columns are arranged by the increasing order of their norms to improve the efficiency of early rejection. Subsequently, the neighbors with smaller d are examined earlier to further optimize the overall complexity.

Algorithm 2 summarizes the proposed QR-TS detection algorithm with the ordering schemes. In Algorithm 2, steps 1–5 and 7–16 correspond to Tx-ordering and Rx-ordering, respectively. When Tx-ordering and Rx-ordering are not applied, $\underline{\boldsymbol{H}}$ and $\underline{\boldsymbol{r}_d}$ are set to the original unordered ones, as indicated in steps 4 and 15, respectively. Steps 17–19 are for the initialization phase, whereas steps 20–38 present the iterative searching phase. In each iteration, the neighbors are ordered for Tx-ordering in step 22. Algorithm 1 is then used

Algorithm 2 QR-TS Detection with Tx- and Rx-ordering

Input: $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N]$, $\boldsymbol{y}$

if Tx-ordering is used **then**
  Find $\underline{\boldsymbol{H}}$ by sorting the columns of $\boldsymbol{H}$ in the ascending order of $\|\boldsymbol{h}_n\|^2$.
else
  $\underline{\boldsymbol{H}} = \boldsymbol{H}$
end if
Obtain $\boldsymbol{Q}$ and $\boldsymbol{R}$ by QR-decomposition of $\underline{\boldsymbol{H}}$.
$\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}$
if Rx-ordering is used **then**
  for $d = 1$ to N **do**
    $\mathcal{S}_d = \{1, \ldots, d\}$
    Find $\underline{\mathcal{S}_d}$ by sorting the elements of $\mathcal{S}_d$ in the descending order of $|r_{n,d}|^2, n = 1, 2, \ldots, d$.
    Find $\underline{\boldsymbol{r}_d}$ by sorting the first d elements of $\boldsymbol{r}_d$ in the order $\underline{\mathcal{S}_d}$.
  end for
else
  $\underline{\boldsymbol{r}_d} = \boldsymbol{r}_d$
end if
$\boldsymbol{x}_{ZF} = \ulcorner \boldsymbol{H}^{\dagger}\boldsymbol{y} \lrcorner$
Initialize $\boldsymbol{c}_{\{1\}}$ based on $\boldsymbol{x}_{ZF}$ and push it to $\mathcal{L}$, $p = 1$ (current length of $\mathcal{L}$).
$\hat{\boldsymbol{s}}_{TS} = \boldsymbol{c}_{\{1\}}, \phi(\hat{\boldsymbol{s}}_{TS}) = \phi(\boldsymbol{c}_{\{1\}}), \boldsymbol{c} = \boldsymbol{c}_{\{1\}}, d^{\star} = N$
for $i = 1$ to $\mathcal{I}$ **do**
  Find the neighbor set $\mathcal{N}(\boldsymbol{c})$ and the corresponding set of difference positions $\mathcal{D}$.
  if Tx-ordering **then**
    Sort the neighbors in the ascending order of the elements of $\mathcal{D}$: $\underline{\mathcal{N}}(\boldsymbol{c}) = \{\boldsymbol{x}_{[1]}, \boldsymbol{x}_{[2]}, \ldots, \boldsymbol{x}_{[L]}\}$.
  else
    $\underline{\mathcal{N}}(\boldsymbol{c}) = \mathcal{N}(\boldsymbol{c})$
  end if
  Sort the first d elements of $\boldsymbol{z}$ in the order $\underline{\mathcal{S}_d}$.
  Use Algorithm 1 to find the best neighbor $\boldsymbol{x}^{\star}$ in the neighbor set $\underline{\mathcal{N}}(\boldsymbol{c})$
  Set $d^{\star}$ to the difference position of $\boldsymbol{x}^{\star}$.
  Move to the best neighbor, $\boldsymbol{c} = \boldsymbol{x}^{\star}$, and update $\boldsymbol{z}$.
  if $\phi(\boldsymbol{c}) < \phi(\hat{\boldsymbol{s}}_{TS})$ **then**
    Update the best solution: $\hat{\boldsymbol{s}}_{TS} = \boldsymbol{c}, \phi(\hat{\boldsymbol{s}}_{TS}) = \phi(\boldsymbol{c})$.
  end if
  if Tabu list $\mathcal{L}$ is full **then**
    Release the first element in $\mathcal{L}$, $p = p - 1$.
  end if
  Push $\boldsymbol{c}$ to $\mathcal{L}$.
  Update the length of the tabu list, $p = p + 1$.
end for
The final solution is the best solution $\hat{\boldsymbol{s}}_{TS}$ found so far.

to find the best neighbor in step 27. It is worth noting that for each neighbor corresponding to d, the elements of $[z_1, z_2, \ldots, z_d]$ are also sorted in the order $\mathcal{S}_d$ to generate $\underline{\boldsymbol{z}}$. The metric is now expressed as

$$\phi(\boldsymbol{x}) = \left\| \underline{\boldsymbol{z}} + \underline{\boldsymbol{r}_d} \delta_d \right\|^2 . \tag{2.14}$$

Finally, after $\mathcal{I}$ searching iterations, the output in step 40 is the best solution found. Henceforth, we refer to the case when both Tx- and Rx-ordering are applied as "double-ordering."

It is also noteworthy that the proposed Tx-ordering scheme has a different objective, procedure, and benefits compared with the ordering scheme used in the LTS algorithm [10]. Specifically, in LTS, post-detection SNR-based ordering is performed to order symbols such that the interference among the layers is minimized. This scheme can improve detection performance, but requires high complexity [10]. In contrast, the proposed Tx-ordering reduces the complexity of the QR-TS algorithm with no loss in performance.

2.3.3. Complexity analysis

In this section, we examine the amount of computation reduction achieved by the proposed QR-TS scheme by comparing it to the conventional algorithm in terms of complexity.

1) Initialization phase

First, we analyze the initialization phase, where the ZF-based candidate $\boldsymbol{c}_{\{1\}}$ is determined. In this work, we assume that the QR Householder method is used to perform QR decomposition, which requires $\frac{2}{3}N^3$ multiplications and $\frac{2}{3}N^3$ additions [43]. In QR-TS, unlike the conventional TS algorithm, the initial solution $\boldsymbol{c}_{\{1\}}$ can be determined as

$\boldsymbol{c}_{\{1\}} = \boldsymbol{x}_{ZF} = \ulcorner \boldsymbol{H}^{\dagger}\boldsymbol{y} \lrcorner = \ulcorner \boldsymbol{R}^{-1}\boldsymbol{Q}^{T}\boldsymbol{y} \lrcorner$. Here, the computation of $\boldsymbol{Q}^{T}\boldsymbol{y}$ requires N^2 multiplications and $N^2 - N$ additions. Because $\boldsymbol{R}$ is an upper triangular matrix, solving for $\boldsymbol{x}_{ZF}$ by performing the backward substitution costs $\frac{1}{2}N^2 + \frac{1}{2}N$ multiplications and $\frac{1}{2}N^2 - \frac{1}{2}N$ additions [43, 44].

The cost metric of $\boldsymbol{c}_{\{1\}}$ is computed as

$$\phi\left(\boldsymbol{c}_{\{1\}}\right) = \left\| \boldsymbol{Q}^{T}\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}_{\{1\}} \right\|^2, \tag{2.15}$$

which requires $\frac{1}{2}N^2 + \frac{3}{2}N$ multiplications and $\frac{1}{2}N^2 + \frac{1}{2}N$ additions, with the note that most operations are to compute $\boldsymbol{R}\boldsymbol{c}_{\{1\}}$ because $\boldsymbol{Q}^{T}\boldsymbol{y}$ is already computed while finding $\boldsymbol{c}_{\{1\}}$. Therefore, the total complexity of the initialization phase of QR-TS is $\frac{2}{3}N^3 + 2N^2 + 2N$ multiplications and $\frac{2}{3}N^3 + 2N^2 - N$ additions.

2) Searching phase

Similar to the computation of $\boldsymbol{u}$ in the conventional TS, the computation of $\boldsymbol{z}$ is only required in the first searching iteration. In the subsequent iterations, $\boldsymbol{z}$ is automatically obtained during the neighbor-searching process. Specifically, $\boldsymbol{z}$ in the $(i+1)$th iteration is expressed as

$$\boldsymbol{z}_{\{i+1\}} = \boldsymbol{Q}^{T}\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}_{\{i+1\}} = \left(\boldsymbol{z} + \boldsymbol{R}\Delta\boldsymbol{x}\right)_{\{i\}}, \tag{2.16}$$

where $\boldsymbol{z}_{\{i\}} = \boldsymbol{Q}^{T}\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}_{\{i\}}$ according to the definition of $\boldsymbol{z}$ in (2.5). In the ith iteration, as an essential procedure to find the best neighbor, $\left(\boldsymbol{z} + \boldsymbol{R}\Delta\boldsymbol{x}\right)_{\{i\}}$ are computed for $L_{\{i\}}$ neighbors, and one of them is $\boldsymbol{z}_{\{i+1\}}$, as shown in (2.16). Therefore, the computational loads for $\boldsymbol{z}$ in the searching phase is ignored in the following analysis.

Complexity in computing ϕ_B: The most intuitive factor for the computational saving of QR-TS is the QR decomposition advantage. Because $\boldsymbol{R}$ is an upper triangular matrix and $\Delta\boldsymbol{x}$ is a single-nonzero vector, the computational complexity for $\boldsymbol{R}\Delta\boldsymbol{x}$ is much lower than that for $\boldsymbol{H}\Delta\boldsymbol{x}$. Furthermore, the zeros in $\boldsymbol{R}\Delta\boldsymbol{x} = \boldsymbol{r}_d\delta_d = [r_{1,d}\delta_d, \ldots, r_{d,d}\delta_d, 0, \ldots, 0]^T$ lead to the low-complexity computation as described in Section 2.3.1, where the computations of $\phi_B = \sum_{n=d+1}^{N} |z_n|^2$ are performed only once for all L neighbors. More specifically, the complexity to find ϕ_B for all neighbors are equal to that to find $\boldsymbol{e}$ in (2.7). According to Fig. 2.1 and steps 2–4 in Algorithm 1, it requires only $N-1$ multiplications and $N-2$ additions.

Average complexity to find the best metric: Another important factor for the low complexity of QR-TS is the early rejection scheme described in Section 2.3.1. We recall that in QR-TS, it takes only $N-1$ multiplications and $N-2$ additions to compute ϕ_B. Furthermore, $\sum_{l=1}^{L} 2\eta_l$ multiplications and $\sum_{l=1}^{L} 2\eta_l - 1$ additions are required to compute the summation in (2.9) for all neighbors corresponding to the vertical paths in Fig. 2.1. Let $\bar{\eta}$ be the average value of $\eta_l, l = 1, 2, \ldots, L$. The average computational cost of QR-TS to find the best neighbor in an iteration are $N - 1 + 2\bar{L}\bar{\eta}$ multiplications and $N - 2 + \bar{L}\,(2\bar{\eta} - 1)$ additions. Furthermore, it can be observed in Algorithm 1 that the QR-TS algorithm requires an average of $\bar{L}(\bar{\eta} + 1)$ comparisons for steps 10 and 16 in each iteration.

In Table 2.3, we summarize and compare the computational complexities of the conventional TS and the proposed QR-TS algorithms. Additionally, we present the upper bounds on their complexities to find the best neighbor. It should be noted that $\bar{L} \leq N - 1$ for BPSK/QPSK and $\bar{L} \leq 2N - \alpha - 1$ for higher-order modulation schemes, such as 16- and 64-QAM, and that $\bar{\eta} \leq \bar{d} \approx \frac{N}{2}$. These analytical results will be verified numerically in Section

2.4. From Table 2.3, it is clear that, in large-sized MIMO systems, the QR-TS algorithm requires a complexity approximately two times lower to find the best neighbor compared with the conventional TS algorithm. Furthermore, the average overall complexities to find the best neighbor in the conventional TS and QR-TS algorithms can be expressed as

$$\mathcal{C}_{TS} = 4\bar{L}N, \tag{2.17}$$

$$\mathcal{C}_{QR-TS} = 2N + 5\bar{\eta}\bar{L} - 3, \tag{2.18}$$

respectively. The upper bounds of $\mathcal{C}_{TS}$ for BPSK/QPSK and higher-order modulation schemes are given by

$$\mathcal{C}_{TS}^{ub,low} = 4N^2 - 4N, \tag{2.19}$$

$$\mathcal{C}_{TS}^{ub,high} = 8N^2 - 4(\alpha + 1)N, \tag{2.20}$$

respectively. On the other hand, the corresponding upper bounds of $\mathcal{C}_{QR-TS}$ are

$$\mathcal{C}_{QR-TS}^{ub,low} = \frac{5}{2}N^2 - \frac{1}{2}N - 3, \tag{2.21}$$

$$\mathcal{C}_{QR-TS}^{ub,high} = 5N^2 - \frac{5\alpha + 1}{2}N - 3. \tag{2.22}$$

It is worth noting that the complexity of QR-TS significantly depends on $\bar{\eta}$ as shown in (2.18). In the QR-TS algorithm, N and $\bar{L}$ can be considered as constants, and $\bar{\eta}$ is the unique factor that decides the overall complexity, which motivates us to develop techniques to reduce $\bar{\eta}$, or equivalently, accelerate the increase of cumulative metrics.

TABLE 2.3. Comparison of the number of multiplications, additions, and comparisons of the conventional TS and QR-TS algorithms

Process		Conventional TS			QR-TS		
		Multiplication	Addition	Comparison	Multiplication	Addition	Comparison
Initialization phase		$N^3/3 + 3N^2/2 + N/6$	$N^3/3 + 3N^2/2 + N/6 - 1$	0	$2N^3/3 + 2N^2 + 2N$	$2N^3/3 + 2N^2 - N$	0
Find the best neighbor in one iteration	Average	$2\bar{L}N$	$\bar{L}(2N-1)$	$\bar{L}$	$N - 1 + 2\bar{L}\bar{\eta}$	$N - 2 + \bar{L}(2\bar{\eta} - 1)$	$\bar{L}(\bar{\eta} + 1)$
	Upper bound (QPSK, BPSK)	$2N^2 - 2N$	$2N^2 - 3N + 1$	$N - 1$	$N^2 - 1$	$N^2 - N - 1$	$N^2/2 + N/2 - 1$
	Upper bound (16-QAM, 64-QAM)	$4N^2 - 2(\alpha+1)N$	$4N^2 - 2(\alpha+2)N + (\alpha+1)$	$2N - \alpha - 1$	$2N^2 - \alpha N - 1$	$2N^2 - (\alpha+2)N + (\alpha-1)$	$N^2 - (\alpha-3)N/2 - (\alpha+1)$

3) Complexity of ordering schemes

In Algorithm 2, the computations for the Tx- and Rx-ordering schemes are performed in steps 2 and 11, respectively. To perform step 2 for Tx-ordering, the norms of the N columns of $\boldsymbol{H}$, i.e., $\|\boldsymbol{h}_n\|^2, n = 1, \ldots, N$ are required. However, for the real channel matrix expressed in (2), we have $\|\boldsymbol{h}_t\|^2 = \left\|\boldsymbol{h}_{t+\frac{N}{2}}\right\|^2, t = 1, \ldots, \frac{N}{2}$. Therefore, only $\frac{N}{2}$ norm values, i.e., $\|\boldsymbol{h}_t\|^2, t = 1, \ldots, \frac{N}{2}$, need to be computed, which requires $\frac{N^2}{2}$ multiplications and $\frac{N^2-N}{2}$ additions. Furthermore, assuming that the quick-sort algorithm [45] is used to sort the channel columns, $\frac{N}{4}(\log_2 N - 1)$ comparisons are required on average. Therefore, the total complexity of Tx-ordering is $N^2 - \frac{3N}{4} + \frac{N}{4}\log_2 N$ operations.

In Rx-ordering, for each d, the computations and sorting of $|r_{n,d}|^2, n = 1, 2, \ldots, d$ in step 11 require d multiplications and $\frac{1}{2}d\log_2 d$ comparisons on average if the quick-sort algorithm [45] is used. For $d = 1, 2, \ldots, N$, the total complexity of Rx-ordering becomes $\sum_{d=1}^{N}(d + \frac{1}{2}d\log_2 d) = \frac{1}{2}(N^2 + N + \log_2 \mathcal{H}(N))$ operations, where $\mathcal{H}(\cdot)$ is the hyperfactorial function [46].

In Algorithm 2, the computations of both Tx- and Rx-ordering are performed outside the iterative searching process of $\mathcal{I}$ iterations. Therefore, although $N^2 - \frac{3N}{4} + \frac{N}{4}\log_2 N$ and $\frac{1}{2}(N^2 + N + \log_2 \mathcal{H}(N))$ operations are required for Tx- and Rx-ordering, respectively, these amounts are much smaller than the total complexities required to find the best neighbors in $\mathcal{I}$ searching iterations. As a result, despite the additional required complexity for ordering, the complexity advantages of the proposed ordering schemes are still significant. This will be numerically verified in the next section.

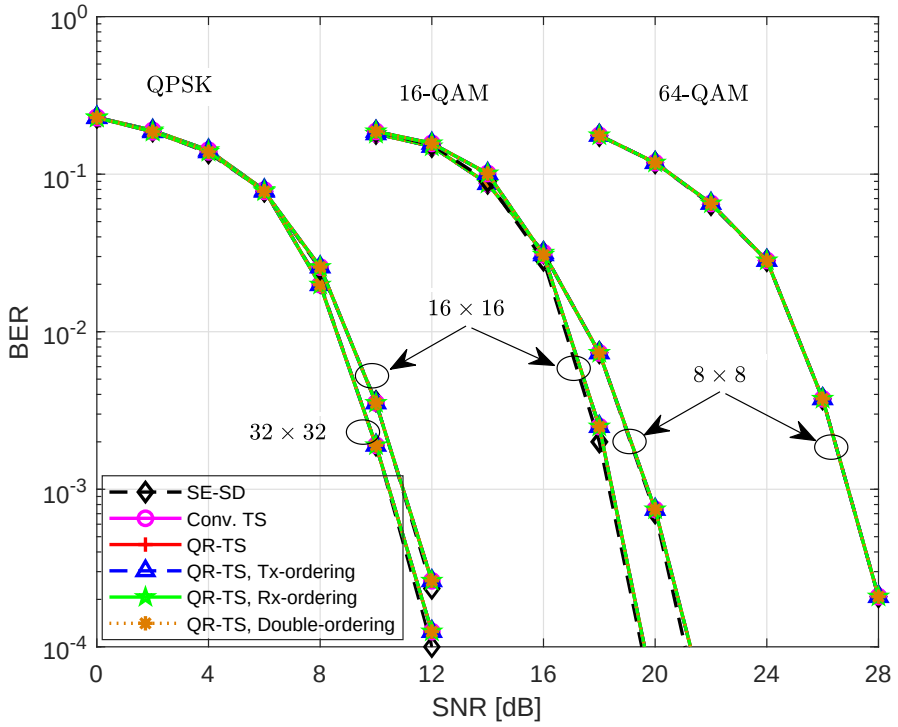


FIG. 2.2. BER performance comparison for 16×16 and 32×32 MIMO with QPSK, 8×8 and 16×16 MIMO with 16-QAM, and 8×8 MIMO with 64-QAM.

2.4. Simulation results

In this section, we numerically evaluate the BER performance and computational complexities of the proposed schemes, which are also compared to those of the conventional TS, LTS [10], Schnorr–Euchner SD (SE-SD) [47], and KSD receivers [5], [7]. For a fair comparison, in the simulations, we used the real-valued signal model for QR-TS, LTS, and conventional TS algorithm. In our simulations, each channel coefficient is assumed to be an i.i.d. zero-mean complex Gaussian random variable with a variance of $1/2$ per dimension. The SNR is defined as the ratio $N_t\sigma_t^2/\sigma_v^2$.

2.4.1. BER performance of the proposed schemes

In Fig. 2.2, we compare the BER performance of the proposed QR-TS algorithms to those of the conventional TS and SE-SD receivers in various MIMO environments, namely, 16×16 and 32×32 MIMO with QPSK, 8×8 and 16×16 MIMO with 16-QAM, and 8×8 MIMO with 64-QAM[1]. For each environment, it is shown that with the chosen $\mathcal{I}$ and P values, which are presented in Table 2.4, the BER performances provided by the conventional TS and QR-TS algorithms are approximately the same as that of the SE-SD decoder. It is also shown that the proposed schemes including the QR-TS and its variations with the ordering schemes totally preserve the BER performance of the conventional TS algorithm.

TABLE 2.4. $\mathcal{I}$, P for QR-TS and the conventional TS to achieve SE-SD performances in various systems, and $\bar{L}$, $\bar{\eta}$ obtained by simulations.

Modulation scheme	$N_t \times N_r$	$[\mathcal{I}, P]$	$\bar{L}$	$\bar{\eta}$
QPSK	2×2	$[6, 3]$	2.5	2.13
	4×4	$[40, 20]$	6.45	3.42
	8×8	$[150, 75]$	14.44	6.12
	16×16	$[400, 200]$	30.50	12.28
	32×32	$[800, 400]$	62.43	25.75
	64×64	$[2000, 1000]$	126.20	56.25
16-QAM	2×2	$[100, 50]$	4.16	2.13
	4×4	$[400, 200]$	11.87	3.38
	8×8	$[2500, 1250]$	27.15	6.04
	16×16	$[8000, 4000]$	57.02	11.98

For the subsequent simulations carried to compare the computational complexity, we

[1]We note that the complexity reduction through effective metric computation, early rejection, and ordering schemes can be achieved for every antenna configuration. However, in this study, we focus on a typical MIMO assumption of $N_r = N_t$, which is generally considered a practical antenna configuration to simultaneously achieve high data rates and relatively low complexities.

chose the values of $\mathcal{I}$ and P such that QR-TS achieves a performance approximately equal to that of SE-SD, at BERs between 10^{-3} and 10^{-4} in each environment. Therefore, the determined values of $\mathcal{I}$ and P guarantee that the considered algorithms require nearly the same SNRs to achieve BERs in the 10^{-3}–10^{-4} range. This ensures a fair comparison between the algorithms in terms of complexity. We note that in the TS algorithm, the length of the tabu list, i.e., P, affects the performance. Specifically, a small P can result in a high chance of cycling in the search, whereas large P can result in a high chance of forbidding necessary moves to obtain the ML solution. Based on the simulations, we observed that the optimal performance is achieved approximately for $P = \mathcal{I}/2$. Therefore, we assume $P = \mathcal{I}/2$ for our simulations, which is also used in [27]. Table 2.4 lists the values of $\mathcal{I}$ and P for each environment, which are used in the simulations to compare the complexity of the algorithms.

2.4.2. Complexity to find the best neighbor

In Section 2.3.3, we showed that the proposed QR-TS algorithm can significantly reduce the complexity required to find the best neighbor via efficient computation and early rejection. In this section, we numerically verify it. First, via simulations, we obtained the average number of neighbors, $\bar{L}$, and the average number of accumulated layers in (2.10), $\bar{\eta}$, for each environment. The values of $\bar{L}$ and $\bar{\eta}$ are presented in Table 2.4 and used to compute the average complexity to find the best neighbor.

In Fig. 2.3, the average complexity required to find the best neighbor in the conventional TS and the proposed QR-TS algorithms are plotted. Specifically, the simulation, analytical results, and upper bounds are compared for the cases of $N_t = \{2, 4, 8, 16, 32, 64\}$ with QPSK and $N_t = \{2, 4, 8, 16\}$ with 16-QAM. In Fig. 2.3, the analytical results and upper

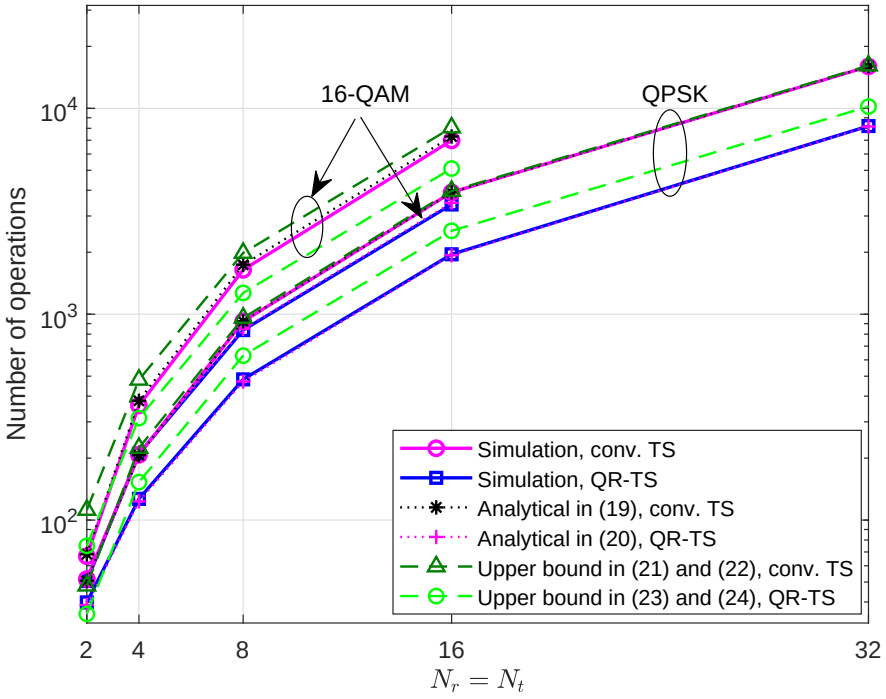


FIG. 2.3. Average computational complexity to find the best neighbor for the cases of $\mathcal{I} = \{6, 40, 150, 400, 800, 2000\}$ for $N_t = \{2, 4, 8, 16, 32, 64\}$, SNR = 12 dB, QPSK, and $\mathcal{I} = \{100, 400, 2500, 8000\}$ for $N_t = \{2, 4, 8, 16\}$, SNR = 22 dB, and 16-QAM. $N_r = N_t$ and $P = \mathcal{I}/2$ are assumed.

bounds for the conventional TS and QR-TS algorithms obtained using (2.17)–(2.22) are shown. Furthermore, to plot the upper bounds of the complexities in (2.20) and (2.22) for 16-QAM, we assume $\alpha = 0$, which results in the highest complexities. The values for $\mathcal{I}, P$, N_t, $\bar{L}$, and $\bar{\eta}$ are chosen from Table 2.4. It is clear that the analytical expressions correctly reflect the computational costs of the considered algorithms. It is also shown that the upper bounds provide close approximations to the computational complexities, especially for QPSK modulation. For 16-QAM modulation, the upper bounds are not as close to the simulation and analytical results because the highest complexity is plotted for 16-QAM, i.e., $\alpha = 0$. In this figure, it is shown that the QR-TS algorithm requires a much lower average complexity to find the best neighbor compared to the conventional TS.

2.4.3. Complexity comparison with conventional TS

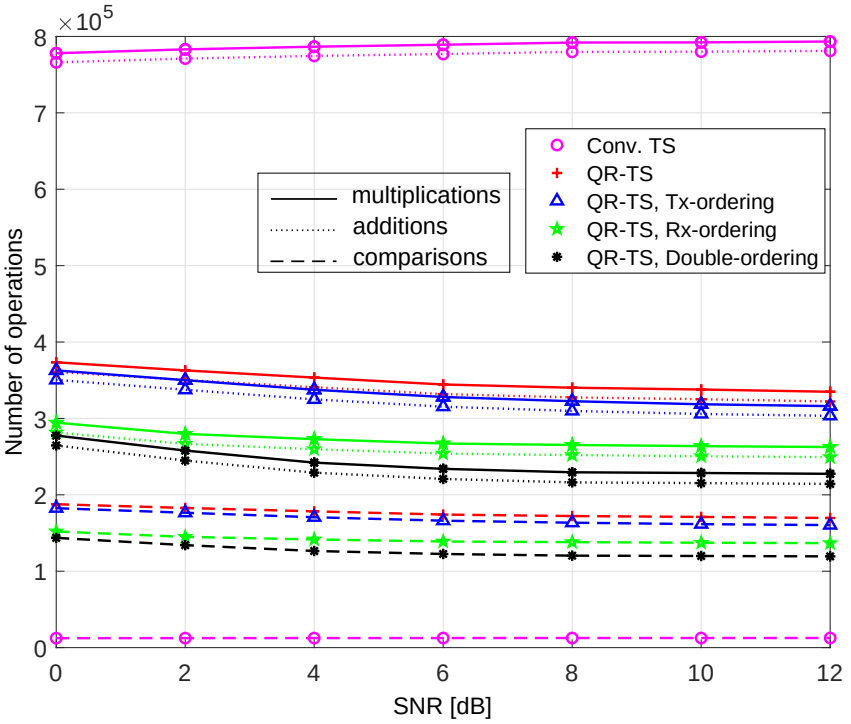
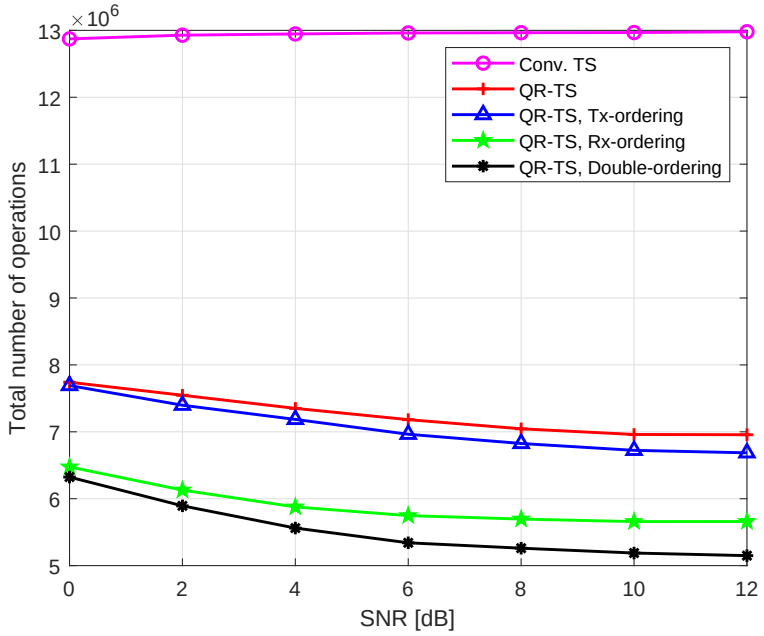


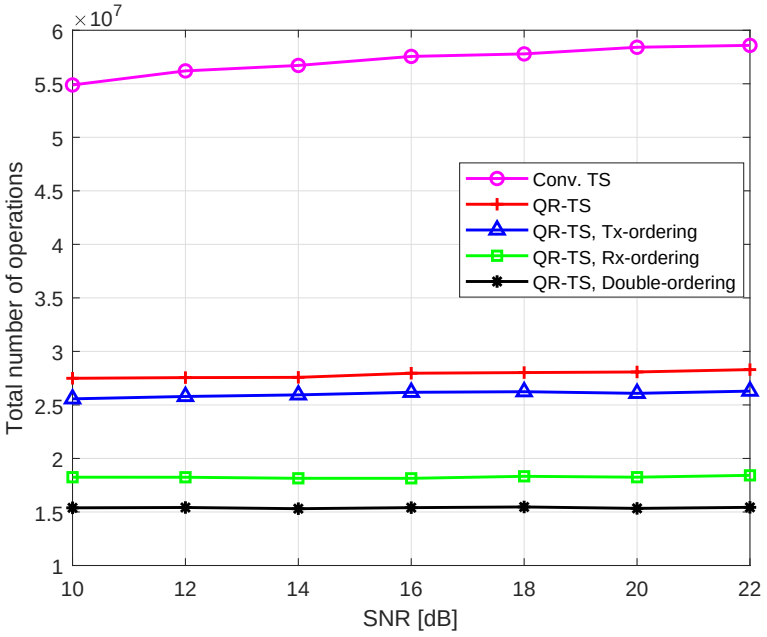
FIG. 2.4. Average numbers of multiplications, additions, and comparisons of the proposed schemes in comparison with those of the conventional TS for 16×16 MIMO, $\mathcal{I} = 400, P = 200$, and QPSK.

In this subsection, the complexities of the proposed QR-TS algorithms are compared with that of the conventional TS algorithm. Fig. 2.4 shows the numbers of additions, multiplications, and comparisons required by the conventional TS algorithm and the proposed schemes. In Fig. 2.4, it can be observed that QR-TS and its variants using the ordering schemes require significantly lower complexities compared with the conventional TS algorithm in terms of the numbers of additions and multiplications involved. It should be noted that although Fig. 2.4 shows that the proposed schemes require a larger number of comparisons than the conventional TS algorithm, its proportion to the overall complexity is small.

For simplicity, in the remaining comparisons of complexity, only the overall complexity, which is the total number of additions, multiplications, and comparisons, is presented. In Fig. 2.5, we compare the overall complexities of the proposed schemes to that of the conventional TS algorithm in various environments, namely, 32×32 MIMO with QPSK and $[\mathcal{I}, P] = [800, 400]$ in Fig. 2.5A, and 16×16 MIMO with 16-QAM and $[\mathcal{I}, P] = [8000, 4000]$ in Fig. 2.5B. From Fig. 2.5, it is clear that the QR-TS is capable of significantly reducing the complexity of the TS algorithm. More specifically, the QR-TS algorithm requires approximately only half the complexity needed for the conventional TS. Furthermore, when ordering techniques are applied, the QR-TS can further reduce the complexity. It is shown that Rx-ordering achieves a greater complexity reduction than Tx-ordering. Furthermore, when double-ordering is employed, the complexity reduction ratio of the QR-TS compared with the conventional TS algorithm is 60.8% and 74.1% for 32×32 MIMO with QPSK modulation and 16×16 MIMO with 16-QAM modulation, respectively. Moreover, in the proposed schemes, the complexity reduction becomes more significant as the SNR increases. The reason is that a higher SNR results in a higher chance of early rejection.



(A) 32×32 MIMO, $\mathcal{I} = 800$, $P = 400$, and QPSK



(B) 16×16 MIMO, $\mathcal{I} = 8000$, $P = 4000$, and 16-QAM

FIG. 2.5. Average computational complexities of the proposed schemes in comparison with that of the conventional TS algorithm.

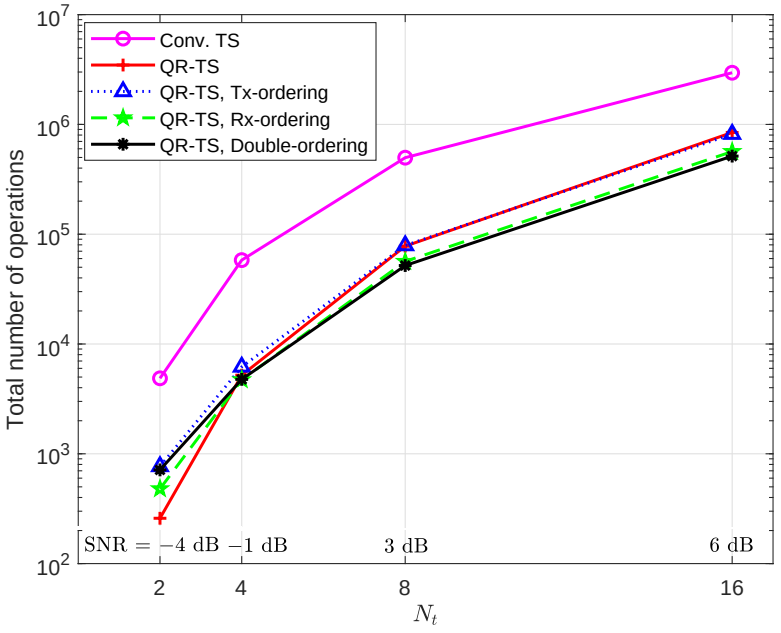
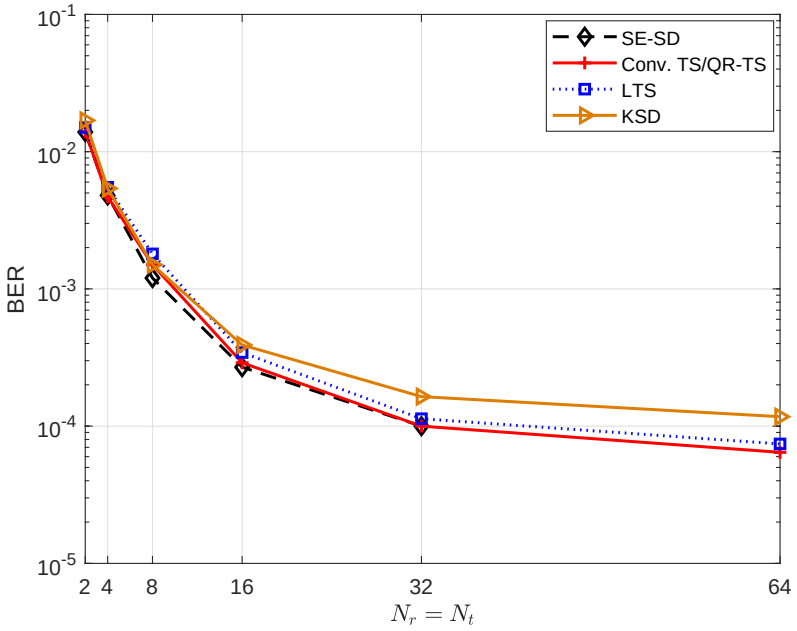
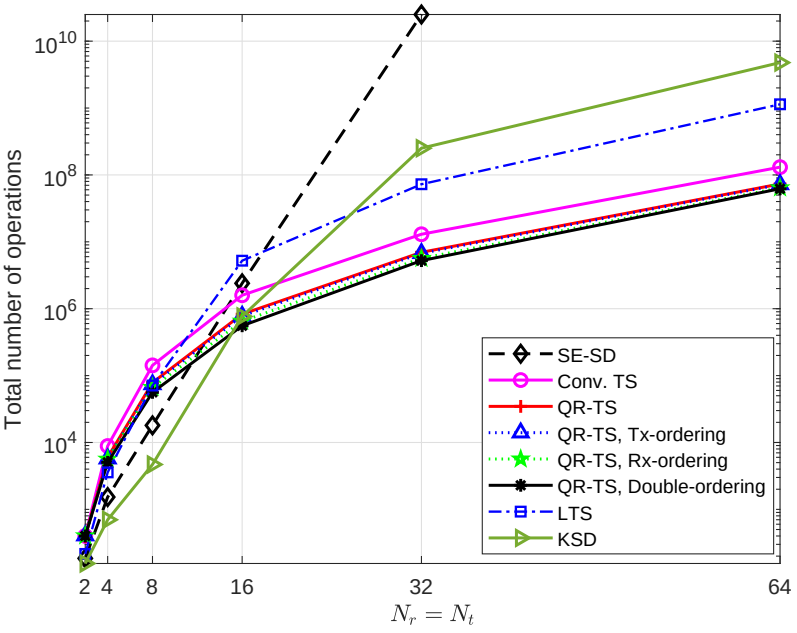


FIG. 2.6. Comparison of complexities of conventional TS and QR-TS for $N_r = 32, N_t = \{2, 4, 8, 16\}$, corresponding to $\mathcal{I} = \{6, 40, 150, 400\}, P = \mathcal{I}/2$, SNR $= \{-4, -1, 3, 6\}$ dB, and QPSK.

The complexity reduction of the proposed schemes for the case $N_r > N_t$ with QPSK is shown in Fig. 2.6 for $N_r = 32$ and $N_t = \{2, 4, 8, 16\}$. It is observed that for $N_r > N_t$, the proposed schemes require significantly lower complexities compared with the conventional TS algorithm. For example, in the case of $(N_t, N_r) = (8, 32)$ with SNR = 3 dB, the QR-TS with double ordering achieves approximately 90% complexity reduction w.r.t. the conventional TS scheme. In the case of $N_t \ll N_r$ such as $(N_t, N_r) = (2, 32)$, the number of neighbors in each searching iteration becomes small. Therefore, the complexity reduction from the ordering cannot compensate for the complexities required for ordering. Consequently, in this case, the QR-TS algorithm without ordering becomes more computationally efficient than that with ordering schemes.



(A) BER performance



(B) Computational complexity

FIG. 2.7. Comparison of the performances and complexities of the conventional TS, QR-TS, LTS, KSD, and SE-SD schemes with QPSK and SNR = 12 dB.

2.4.4. Complexity comparison with the LTS, SE-SD, and KSD

Finally, we compare the computational complexities of the proposed schemes with those of the LTS [10], SE-SD [47], and KSD schemes [5], [7] as well as with conventional TS. We note that SE-SD and KSD are two of the major variants of the original SD algorithm, which are capable of significantly reducing the complexity of SD while achieving the near-optimal performance [5, 47]. Among the variants of TS-based detection, LTS, which also uses QR-decomposition, is chosen for comparison to show the different effects of QR-decomposition to the complexity reduction in LTS and the proposed QR-TS algorithm. The comparisons are made for various environments corresponding to $N_t = N_r \in \{2, 4, 8, 16, 32, 64\}$, with QPSK modulation. For these simulations, we set SNR = 12 dB, for which all the examined algorithms achieve BERs between 10^{-3} and 10^{-4} in large MIMO systems. To ensure a fair comparison in terms of complexity, we first performed simulations to find the appropriate parameters for the LTS and KSD algorithms so that they provide approximately the same BER performances as the TS, QR-TS, and SE-SD algorithms.

It should be noted that QR-TS and its ordering schemes totally preserve the performance of the conventional TS scheme, as shown in Fig. 2.2. Therefore, in Fig. 2.7A, only a single curve is plotted to present their performances. For LTS, the skipping criterion δ is assumed to be 0.25, as in [10]. For $N_t = N_r \in \{2, 4, 8, 16, 32, 64\}$, the number of searching iterations is set to $\mathcal{I}_{LTS} = \{4, 10, 30, 280, 500, 1000\}$, respectively, while the value of K in KSD [5], [7] is chosen to be $K = \{3, 4, 10, 10^4, 10^5, 5 \times 10^5\}$, respectively. Because of the extremely high complexity of the SE-SD scheme for $N_t = 64$, which makes it infeasible to numerically test it in a reasonable time, SE-SD is not examined for $N_t = 64$. In Fig. 2.7A, it can be seen that under these assumptions for the parameters the performance of the KSD and LTS algorithms is close to that of the SE-SD and QR-TS algorithms.

In Fig. 2.7B, the computational complexities of the algorithms is compared under the constraint that the compared algorithms must achieve approximately the same BER performance, as shown in Fig. 2.7A. From Fig. 2.7B, the following observations can be made:

- The SE-SD and KSD receivers have relatively low complexities for small values of N_t, which implies that they are more suitable for small MIMO systems. However, for $N_t \geq 16$, QR-TS and its ordered versions require significantly lower complexities compared with those required by the SE-SD and KSD schemes.
- The complexity of LTS is comparable to that of QR-TS for small values of N_t. However, because joint TS detection can be required in each layer detection in LTS, its computational burden becomes substantially higher in large MIMO systems than those of the conventional TS and QR-TS algorithms. This is in good agreement with the results presented in [10].
- Among the compared schemes, the proposed QR-TS algorithms require lower complexities than the other schemes for large values of N_t. In particular, the QR-TS algorithm with double-ordering exhibits the lowest complexity.

2.5. Conclusion

In this chapter, we introduced a low-complexity symbol detection algorithm called QR-TS for massive MIMO systems. The QR-TS algorithm is capable of significantly reducing the complexity of the TS algorithm using low-complexity metric computation and early rejection schemes, which are based on the QR decomposition of the channel matrix. The simulation and analytical results show that QR-TS reduces the complexity of the TS algorithm approximately by half without any performance loss. To further optimize the

QR-TS algorithm, we exploit the ordering schemes, namely, Tx-ordering and Rx-ordering. The Tx-ordering scheme rearranges the channel columns and neighbors based on channel column norms, which helps to reduce the threshold γ for early rejection more quickly. In the computation of a neighbor metric, the Rx-ordering scheme considers layers with larger average metrics sooner, allowing the cumulative metric to reach the threshold γ more rapidly. The numerical results show that the QR-TS in combination with the two ordering schemes require only approximately one-fourth the complexity of the conventional TS. It is also shown that the proposed algorithms achieve substantial complexity reduction for both lower-and higher-order modulation schemes, and the complexity reduction w.r.t. the SD decoders, such as SE-SD and KSD, becomes more significant in larger MIMO systems. We note that the proposed QR-TS scheme and existing TS-based algorithms such as LTS [10], R3TS [26], and TS with early termination [27], [28] are not mutually exclusive. In other words, QR-TS can be employed for these algorithms to perform the local search process in a more computationally efficient manner.

3. Groupwise Neighbor Examination for Tabu Search Detection in Large MIMO systems

3.1. Motivation and contribution

In an $N_t \times N_r$ MIMO system with QPSK modulation, the number of neighbors in each searching iteration of TS algorithms can be up to $2N_t - 1$ for QPSK, and $4N_t - 1$ for 16- and 64-QAM [35]. Furthermore, a large MIMO system requires a large number of searching iterations to achieve near-ML performance. Therefore, the overall complexity of the TS-based detection algorithms becomes extremely high in large MIMO systems, and the most complexity arises from determining the best neighbors. This motivates the proposal of a novel TS detection algorithm, called neighbor-grouped TS (NG-TS) in this chapter. Our main contributions can be summarized as follows:

- By expanding the ML cost function, which is a function of a channel column vector, we show that among the neighbors corresponding to the same column norm of a channel matrix, the best one can be determined using a simplified cost function. This requires considerably less complexity than the scheme that employs the conventional

cost function.

- By employing the simplified cost function, we develop the groupwise neighbor examination scheme for the TS algorithms. Specifically, the neighbors are divided into groups, and the groups' best neighbors are compared to determine the final best neighbor. This scheme allows the best neighbor in each iteration to be found with much lower complexity than that of the sequential neighbor-examination approach used in prior TS schemes.

- Based on the complexity analysis of the NG-TS algorithm, we propose a channel ordering scheme for further complexity reduction. Our simulation results show that the proposed schemes can significantly reduce the complexity of the TS algorithm while fully preserving its BER performance. As a result, the performance–complexity tradeoff is improved.

The rest of this chapter is organized as follows: In Section 3.2, the proposed NG-TS scheme is presented. In Section 3.3, the simulation results and numerical discussions are presented. Finally, the conclusions are drawn in Section 3.4.

3.2. Proposed NG-TS algorithm

3.2.1. NG-TS algorithm

For the efficient computation of neighbors' metrics, a reduced cost function has been introduced in Chapter 2:

$$\phi(\boldsymbol{x}) = \|\boldsymbol{z} + \boldsymbol{r}_d \delta_d\|^2 , \tag{3.1}$$

where $\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}$, the unitary matrix $\boldsymbol{Q}$, and upper triangular matrix $\boldsymbol{R}$ are obtained by the QR decomposition of $\boldsymbol{H}$, i.e., $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}$, and $\boldsymbol{r}_d$ is the dth column of $\boldsymbol{R}$. In this work, we employ (3.1) to derive the proposed NG-TS algorithm.

1) Neighbor grouping

In the proposed NG-TS algorithm, the neighbor set $\mathcal{N}(\boldsymbol{c})$ is divided into groups, and the best neighbor of each group is determined based on a simplified cost metric function. Let $\boldsymbol{x}_l$ and d_l be the lth neighbor in $\mathcal{N}(\boldsymbol{c})$ and its difference position, $l = 1, 2, \ldots, L$, where L is the number of neighboring vectors in $\mathcal{N}(\boldsymbol{c})$. We note that the distances between the candidate $\boldsymbol{c}$ to all neighbors are the same, i.e., $|\delta| = |\delta_{d_1}| = \ldots = |\delta_{d_L}|$. By expanding $\phi(\boldsymbol{x}_l)$ in (3.1), we obtain

$$\phi(\boldsymbol{x}_l) = \|\boldsymbol{z}\|^2 + |\delta|^2 \|\boldsymbol{r}_{d_l}\|^2 + 2\delta_{d_l}\boldsymbol{z}^T\boldsymbol{r}_{d_l}. \tag{3.2}$$

It is observed from (3.2) that among the neighbors having the same value for $\|\boldsymbol{r}_{d_l}\|$, the one with smallest $\delta_{d_l}\boldsymbol{z}^T\boldsymbol{r}_{d_l}$ has the smallest ML metric. Let $\mathcal{G}_k$ be a group of neighbors having the same value $\|\boldsymbol{r}_{d_l}\|$, i.e.,

$$\mathcal{G}_k = \{\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{c}) : \|\boldsymbol{r}_{d_i}\| = \eta_k\}, \tag{3.3}$$

where η_k is one of the column norms of $\boldsymbol{R}$, i.e., $\eta_k \in \{\|\boldsymbol{r}_1\|, \|\boldsymbol{r}_2\|, \ldots, \|\boldsymbol{r}_N\|\}$. Hence, the best neighbor $\boldsymbol{x}^\star_{\mathcal{G}_k}$ in group $\mathcal{G}_k$ can be found with a simplified cost function as follows:

$$\boldsymbol{x}^\star_{\mathcal{G}_k} = \arg\min_{\boldsymbol{x}\in\mathcal{G}_k} \{\text{sign}(\delta_{d_l})\gamma_{d_l}\}, \tag{3.4}$$

where $\gamma_{d_l} = \boldsymbol{z}^T\boldsymbol{r}_{d_l}$.

The number of neighboring vectors in each group can be obtained from (3.3), with the note that in the real signal model, $\|\boldsymbol{r}_n\| = \|\boldsymbol{r}_{n+N_t}\|, n = 1, \ldots, N_t$, and $\boldsymbol{x}_n$ and $\boldsymbol{x}_j$ are in the same group when $d_n = d_j$. In QPSK, because each symbol in the alphabet $\{-1, 1\}$ only has one neighboring symbol, each group has a maximum of two vectors $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n+N_t}$. By contrast, in higher-order modulation schemes such as 16-QAM and 64-QAM, whose alphabets are $\{-3, -1, 1, 3\}$ and $\{-7, -5, -3, -1, 1, 3, 5, 7\}$, respectively, each symbol has at most two neighboring symbols. Therefore, there is a maximum of four neighbors in each group, including two pairs of neighbors with the same difference positions. For example, with $M = 4$ and 64-QAM modulation, a group can be formed by four vectors

$$\left\{ \begin{bmatrix} -7 \\ -7 \\ 5 \\ 5 \end{bmatrix}, \begin{bmatrix} -7 \\ -3 \\ 5 \\ 5 \end{bmatrix}, \begin{bmatrix} -7 \\ -5 \\ 5 \\ 7 \end{bmatrix}, \begin{bmatrix} -7 \\ -5 \\ 5 \\ 3 \end{bmatrix} \right\},$$

which are a subset of the neighbor set of $\boldsymbol{c} = [-7, -5, 5, 5]^T$ with the difference positions $\{2, 4\}$.

2) Complexity of finding the groups' best neighbors

Determining the best neighbor for each group requires the computation of $\gamma_{d_l} = \boldsymbol{z}^T \boldsymbol{r}_{d_l} = \sum_{n=1}^{N} z_n r_{n,d_l}$. However, its complexity is less than that of a multiplication between two $N-$element vectors. This is because over two successive searching iterations, only a subset of elements of $\boldsymbol{z}$ is updated as follows:

$$\begin{aligned} \boldsymbol{z}_{\{i+1\}} &= \boldsymbol{Q}^T \boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}_{\{i+1\}} = \boldsymbol{Q}^T \boldsymbol{y} - \boldsymbol{R}\left(\boldsymbol{c}_{\{i\}} - \Delta \boldsymbol{x}_{\{i\}}\right) \\ &= \boldsymbol{z}_{\{i\}} + \boldsymbol{R}\Delta \boldsymbol{x}_{\{i\}} = \boldsymbol{z}_{\{i\}} + \boldsymbol{r}_{d^\star}\delta_{d^\star} \end{aligned}$$

$$= \boldsymbol{z}_{\{i\}} + \left[r_{1,d^\star}\delta_{d^\star}, \ldots, r_{d^\star,d^\star}\delta_{d^\star}, 0, \ldots, 0\right]^T, \tag{3.5}$$

where the subscript $\{i\}$ represents the ith iteration[1], and $\Delta\boldsymbol{x}_{\{i\}} = \boldsymbol{c}_{\{i\}} - \boldsymbol{c}_{\{i+1\}} = [0, \ldots, 0, \delta_{d^\star}, 0, \ldots, 0]^T$ only has one non-zero element at the $d^\star$th position because $\boldsymbol{c}_{\{i\}}$ and $\boldsymbol{c}_{\{i+1\}}$ are neighbors of each other. It is observed from (3.5) that over two successive iterations, only the first $d^\star$ elements of $\boldsymbol{z}$ need to be updated. Then, γ_{d_l} can be computed as

$$\gamma_{d_l} = \boldsymbol{z}^T\boldsymbol{r}_{d_l} = \begin{cases} \tilde{\gamma}_{d_l} + \sum_{n=1}^{d^\star} z_n r_{n,d_l}, d_l \geq d^\star \\ \sum_{n=1}^{d_l} z_n r_{n,d_l}, d_l < d^\star \end{cases}, \tag{3.6}$$

where $\tilde{\gamma}_{d_l} = \sum_{n=d^\star+1}^{d_l} z_n r_{n,d_l}$ was already computed in the previous iteration. Therefore, the computational complexity required in (3.6) to examine a neighbor is only $\min\{d_l, d^\star\}$ multiplications and $\min\{d_l - 1, d^\star\}$ additions.

3) Channel ordering

The complexity to find the best neighbor of each group can be further reduced by statistically minimizing $d^\star$ in searching iterations because the complexity of (3.6) increases with $\min\{d_l, d^\star\}$ and $\min\{d_l - 1, d^\star\}$. We note that the average metric of $\boldsymbol{x}^\star$ can be expressed as $\mathbb{E}\{\phi(\boldsymbol{x}^\star)\} = N\sigma_v^2 + \delta^2\|\boldsymbol{h}_{d^\star}\|^2$ [35], where $d^\star$ is not only the column index but also the difference position of $\boldsymbol{x}^\star$. Therefore, if we order the channel matrix such that $\|\boldsymbol{h}_1\|^2 \leq \ldots \leq \|\boldsymbol{h}_N\|^2$, there is a larger chance that the best neighbor has a small difference position. As a result, $d^\star$ can decrease. This motivates the ordering of channel columns in the increasing order of their norms to reduce the computational complexity of computing γ_d in (3.6).

4) Finding the final best neighbor

[1]For notational convenience, we omit the interation index i if it does not cause any confusion.

Algorithm 3 NG-TS Detection

Input: $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N]$, $\boldsymbol{y}$
Output: $\hat{\boldsymbol{s}}_{TS}$.

1: Obtain $\boldsymbol{Q}$ and $\boldsymbol{R}$ by QR-decomposition of $\boldsymbol{H}$.
2: Compute $f_m = |\delta|^2 \|\boldsymbol{r}_m\|^2, m = 1, 2, \ldots, M$.
3: Order the columns of $\boldsymbol{H}$ in increasing order of $f_m, m = 1, 2, \ldots, M$, if channel ordering is applied.
4: $\boldsymbol{x}_{ZF} = \ulcorner \boldsymbol{R}^{-1}\boldsymbol{Q}^T\boldsymbol{y} \urcorner$
5: $\boldsymbol{c} = \boldsymbol{x}_{ZF}, \boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{c}$
6: $\hat{\boldsymbol{s}}_{TS} = \boldsymbol{c}, \phi(\hat{\boldsymbol{s}}_{TS}) = \phi(\boldsymbol{c}) = \|\boldsymbol{z}\|^2$
7: Push $\boldsymbol{c}$ to the tabu list.
8: **for** $i = 1$ to $\mathcal{I}$ **do**
9: Find the neighbor set $\mathcal{N}(\boldsymbol{c}) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ and the set of difference positions $\mathcal{D} = \{d_1, \ldots, d_L\}$.
10: Divide $\mathcal{N}(\boldsymbol{c})$ into groups $\mathcal{G}_1, \ldots, \mathcal{G}_K$ based on (3.3).
11: **for** $k = 1 \to K$ **do**
12: **for** $l = 1 \to L_k$ **do**
13: Set $\boldsymbol{x}$ and d to the lth neighbor in $\mathcal{G}_k$ and its difference position.
14: $\alpha_l = \text{sign}(\delta_d)\gamma_d$, where $\gamma_d = \boldsymbol{z}^T\boldsymbol{r}_d$
15: **end for**
16: $\hat{l} = \arg\min_l \{\alpha_l\}$
17: Set $\boldsymbol{x}^\star_{\mathcal{G}_k}$ to the $\hat{l}$th neighbor in $\mathcal{G}_k$.
18: Set $d^\star_k$ to the difference position of $\boldsymbol{x}^\star_{\mathcal{G}_k}$.
19: $\beta_k = 2\delta_{d^\star_k}\gamma_{d^\star_k} + f_{d^\star_k}$
20: **end for**
21: $\hat{k} = \arg\min_k \{\beta_k\}$
22: $\boldsymbol{x}^\star = \boldsymbol{x}^\star_{\mathcal{G}_{\hat{k}}}$
23: Move to the best neighbor, $\boldsymbol{c} = \boldsymbol{x}^\star$, and update $\boldsymbol{z}$.
24: Update $\hat{\boldsymbol{s}}_{TS} = \boldsymbol{c}, \phi(\hat{\boldsymbol{s}}_{TS}) = \phi(\boldsymbol{c})$ if $\phi(\boldsymbol{c}) < \phi(\hat{\boldsymbol{s}}_{TS})$.
25: Release the first element in the tabu list if it is full.
26: Push $\boldsymbol{c}$ to the tabu list and update its length.
27: **end for**
28: The final solution is the best solution $\hat{\boldsymbol{s}}_{TS}$ found so far.

Let K be the number of groups of neighbors, and let $\boldsymbol{x}^\star_{\mathcal{G}_k}$ and $d^\star_k$ be the best neighbors in group $\mathcal{G}_k$ and its difference position, respectively. Once $\boldsymbol{x}^\star_{\mathcal{G}_1}, \ldots, \boldsymbol{x}^\star_{\mathcal{G}_K}$ are found, the final best neighbor is set to $\boldsymbol{x}^\star = \boldsymbol{x}^\star_{\mathcal{G}_{\hat{k}}}$ such that

$$\begin{aligned}\hat{k} &= \arg\min_k \phi(\boldsymbol{x}^\star_{\mathcal{G}_k}) \\ &= \arg\min_k \left\{ \|\boldsymbol{z}\|^2 + 2\delta_{d^\star_k}\gamma_{d^\star_k} + |\delta|^2 \left\| \boldsymbol{r}_{d^\star_k} \right\|^2 \right\}\end{aligned}$$

$$= \arg\min_{k} \left\{ 2\delta_{d_k^\star} \gamma_{d_k^\star} + |\delta|^2 \left\| \boldsymbol{r}_{d_k^\star} \right\|^2 \right\}, \tag{3.7}$$

where the last equation is obtained by the fact that $\|\boldsymbol{z}\|^2$ is the same for all neighbors in each searching iteration. We note that $\gamma_{d_k^\star}$ in (3.7) was already computed to search for the groups' best neighbors. Furthermore, $|\delta|^2 \left\| \boldsymbol{r}_{d_k^\star} \right\|^2$ in (3.7) only depends on the constant $|\delta|^2$ and a column of $\boldsymbol{R}$, which remain unchanged over $\mathcal{I}$ searching iterations. Therefore, $|\delta|^2 \left\| \boldsymbol{r}_{d_k^\star} \right\|^2$ needs to be computed only once outside the searching iterations. As a result, the computational complexity to determine the final best neighbor is relatively low.

The proposed NG-TS algorithm is summarized in Algorithm 3. In step 1, matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are obtained by the QR decomposition of $\boldsymbol{H}$. In step 2, $f_m = |\delta|^2 \|\boldsymbol{r}_m\|^2, m = 1, 2, \ldots, M$, are computed to be used in steps 3, 10, and 19, noting that $\|\boldsymbol{r}_m\|^2 = \|\boldsymbol{h}_m\|^2, m = 1, 2, \ldots, M$. Step 4 computes the initial solution $\boldsymbol{x}_{ZF} = \ulcorner \boldsymbol{H}^\dagger \boldsymbol{y} \urcorner$. Then, steps 5–7 assign $\boldsymbol{x}_{ZF}$ to the current candidate $\boldsymbol{c}$, compute $\boldsymbol{z}$, and initialize the solution $\hat{\boldsymbol{s}}_{TS}$, which is then pushed to the tabu list. In step 10, $\mathcal{N}(\boldsymbol{c})$ is divided into K groups of L_k, $k = 1, \ldots, K$, neighboring vectors, which allows finding K groups' best neighbors with a simplified cost function in steps 16 and 17. Then, the best neighbors of the K groups are compared to determine the final best neighbor in steps 20 and 21. The following steps are for updating the best solution and the tabu list, and then conclude the final solution after $\mathcal{I}$ searching iterations.

5) Computational complexity of the NG-TS detection algorithm

The computational complexity of the proposed NG-TS detection algorithm is presented in Table 3.1 with the assumption $M = N$. We assume that the QR Householder method is used to perform QR decomposition in step 1 of Algorithm 3. To solve $\boldsymbol{x}_{ZF}$ in step 4, the backward substitution method is used. In step 14, the computation of γ_d requires

TABLE 3.1. Computational complexity of Algorithm 3

Step	Number of multiplications	Number of additions
1	$2N^3/3$	$2N^3/3$
2	$N^2/4+1$	$N^2/4-N/2$
4	$N^2/2+N/2$	$N^2/2-N/2$
5	$3N^2/2+N/2$	$3N^2/2-N/2$
6	N	$N-1$
12–20	$2K+\sum_{k=1}^{K}\sum_{l=1}^{L_k}(1+\eta)$	$K+\sum_{k=1}^{K}\sum_{l=1}^{L_k}\eta$

$\min\{d_l, d^\star\}$ multiplications and $\min\{d_l-1, d^\star\}$ additions, as discussed in Section 3.2.1. For simplicity, we assume $\min\{d_l-1, d^\star\} \approx \min\{d_l, d^\star\} = \epsilon$. Therefore, computing α_l requires $1+\epsilon$ multiplications and ϵ additions. Furthermore, computing β_k in step 19 requires only 2 multiplications and 1 addition because $f_{d_{k^\star}}$ is already computed in step 2. Therefore, the total complexity of steps 12–20 is $2K+\sum_{k=1}^{K}\sum_{l=1}^{L_k}(1+\epsilon)$ multiplications and $K+\sum_{k=1}^{K}\sum_{l=1}^{L_k}\epsilon$ additions. Then, the average complexity required in an iteration of the NG-TS algorithm can be expressed as $\mathcal{C}_{iter} \approx 3\bar{K}+\bar{L}+2\varepsilon$, where $\bar{K}$ and $\bar{L}$ are the average values of K and L over all iterations, respectively, and

$$\begin{aligned}\varepsilon &= \mathbb{E}\left\{\sum_{k=1}^{K}\sum_{l=1}^{L_k}\epsilon\right\} = \sum_{k=1}^{K}\sum_{l=1}^{L_k}\mathbb{E}\{\epsilon\} \\ &= \bar{L}\mathbb{E}\{\epsilon\} = \bar{L}\sum_{i=1}^{N} i\left(F_\epsilon[i]-F_\epsilon[i-1]\right). \end{aligned} \tag{3.8}$$

Here, $\mathbb{E}\{\cdot\}$ represents an expected value, and $F_X[\cdot]$ denotes the cumulative distribution function (CDF) of a random variable X. Recall that $\epsilon = \min\{d_l, d^\star\}$ with both d_l and $d^\star$ being independent discrete random variables uniformly distributed on $[1, N]$. We have $F_{d_l}[i] = F_{d^\star}[i] = \frac{i}{N}$. Hence, $F_\epsilon[i]$ can be expressed as

$$F_\epsilon[i] = \mathbb{P}\{\min\{d_l, d^\star\} \le i\} = 1-(1-F_{d_l}[i])(1-F_{d^\star}[i])$$

$$= 1 - \left(1 - \frac{i}{N}\right)^2 = \frac{i(2N-i)}{N^2}, \tag{3.9}$$

where $\mathbb{P}\{\cdot\}$ denotes a probability. From (3.8) and (3.9), we have $\varepsilon = \bar{L}\sum_{i=1}^{N}\frac{(2N+1)i-2i^2}{N^2} = \bar{L}\left(\frac{N}{3}+\frac{1}{2}+\frac{1}{6N}\right)$, and $\mathcal{C}_{\text{iter}}$ can be expressed as

$$\mathcal{C}_{\text{iter}} \approx 3\bar{K} + 2\bar{L} + \frac{2\bar{L}N}{3} + \frac{\bar{L}}{3N}. \tag{3.10}$$

From Table 3.1 and (3.10), the overall complexity of the NG-TS algorithm is given as

$$\mathcal{C}_{\text{NG-TS}} \approx \underbrace{\frac{4N^3}{3} + \frac{9N^2}{2} + \frac{3N}{2}}_{\text{initialization}} + \underbrace{\mathcal{I}\left(3\bar{K} + 2\bar{L} + \frac{2\bar{L}N}{3} + \frac{\bar{L}}{3N}\right)}_{\text{iterative search}}. \tag{3.11}$$

Whereas, the overall complexity of the conventional TS algorithm, including the complexity involved in the initialization and iterative searching process, can be given as [36]

$$\mathcal{C}_{\text{Conv. TS}} = \underbrace{\frac{2N^3}{3} + 3N^2 + \frac{N}{3}}_{\text{initialization}} + \underbrace{\mathcal{I}(4\bar{L}N - 2)}_{\text{iterative search}}. \tag{3.12}$$

By comparing (3.11) to (3.12), it is observed that the complexity required for the initialization of the NG-TS algorithm is greater than that of the conventional TS algorithm. However, in large MIMO systems, $\mathcal{I} \gg N$ is required to achieve near-optimal performance [35]. Therefore, the complexity required in the iterative searching process dominates the overall complexity. Furthermore, we note that $\bar{L} \leq N-1$ and $\bar{K} \leq 2$ for BPSK/QPSK, and $\bar{L} \leq 2N-1$ and $\bar{K} \leq 4$ for higher-order modulation schemes, such as 16- and 64-QAM. Therefore, from (3.12) and (3.11), it is clear that the complexity of the proposed NG-TS algorithm is significantly lower than that of the conventional TS algorithm. This will be numerically verified in the next section.

3.3. Simulation results

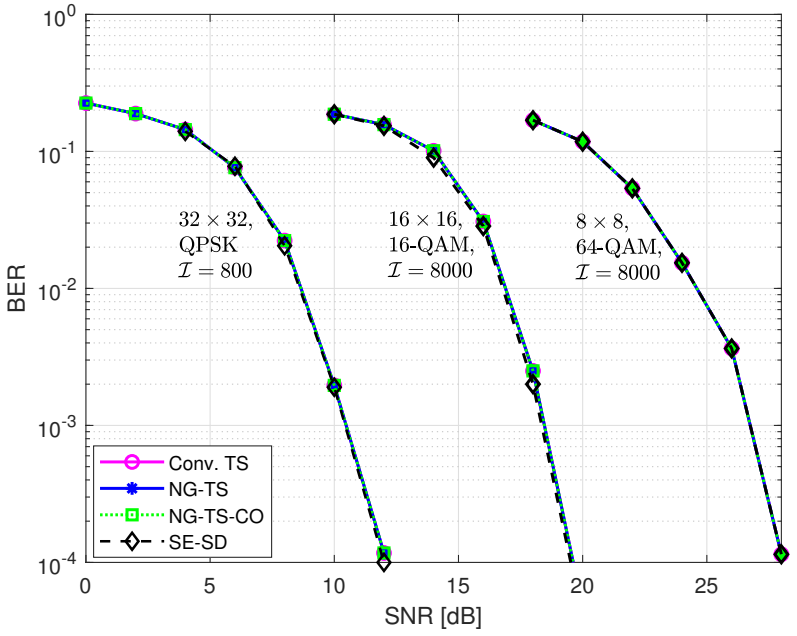


FIG. 3.1. BER performance of the proposed schemes in comparison with those of the conventional TS and SE-SD.

In our simulations, channel coefficients are randomly generated as i.i.d. complex Gaussian random variables with zero mean and variance of $1/2$ per dimension, and SNR is set to $N_t\sigma_t^2/\sigma_v^2$. In both Figs. 3.1 and 3.2, we consider $N_t = N_r = \{32, 16, 8\}$ with QPSK, 16-QAM, and 64-QAM, respectively. For those systems, $\mathcal{I}$ is set to $800, 8000$, and 8000, respectively, which guarantees that the conventional TS and NG-TS algorithms perform approximately the same as the SE-SD decoder [47]. Furthermore, the length of the tabu list is set to $P = \mathcal{I}/2$ so that TS algorithms achieve approximately the optimal performance [27,35].

In Fig. 3.1, we compare the BER performance of the proposed NG-TS algorithms to

those of the conventional TS and SE-SD algorithm. It is shown that in all the three considered systems, the proposed NG-TS and NG-TS with channel ordering (NG-TS-CO) schemes totally preserve the BER performance of the conventional TS algorithm. Furthermore, with the chosen values of $\mathcal{I}$ and P, the performances of TS algorithms are close to those of SE-SD.

In Fig. 3.2, the computational complexities of the NG-TS, conventional TS, QR-TS, and QR-TS with channel ordering (QR-TS-CO) algorithms are compared. The same values of $\mathcal{I}$ and P as in Fig. 3.1 are used. It is shown that in all the considered environments, the complexity reduction ratios of NG-TS and NG-TS-CO compared with the conventional TS algorithm are approximately 80% and 85%, respectively. Furthermore, the complexities of NG-TS are approximately $55\% - 70\%$ lower than those of QR-TS.

It has been shown in [35] that the QR-TS totally preserves the performance of the conventional TS and achieves the performances of SE-SD [47], LTS [10], and KSD [5] algorithms with considerably lower complexities. Therefore, based on the comparisons of the performances and complexities of the conventional TS, QR-TS, and NG-TS in Figs. 3.1 and 3.2, it is clear that the proposed NG-TS achieves the improved performance-complexity trade-off compared to the conventional TS, QR-TS, SE-SD, LTS, and KSD.

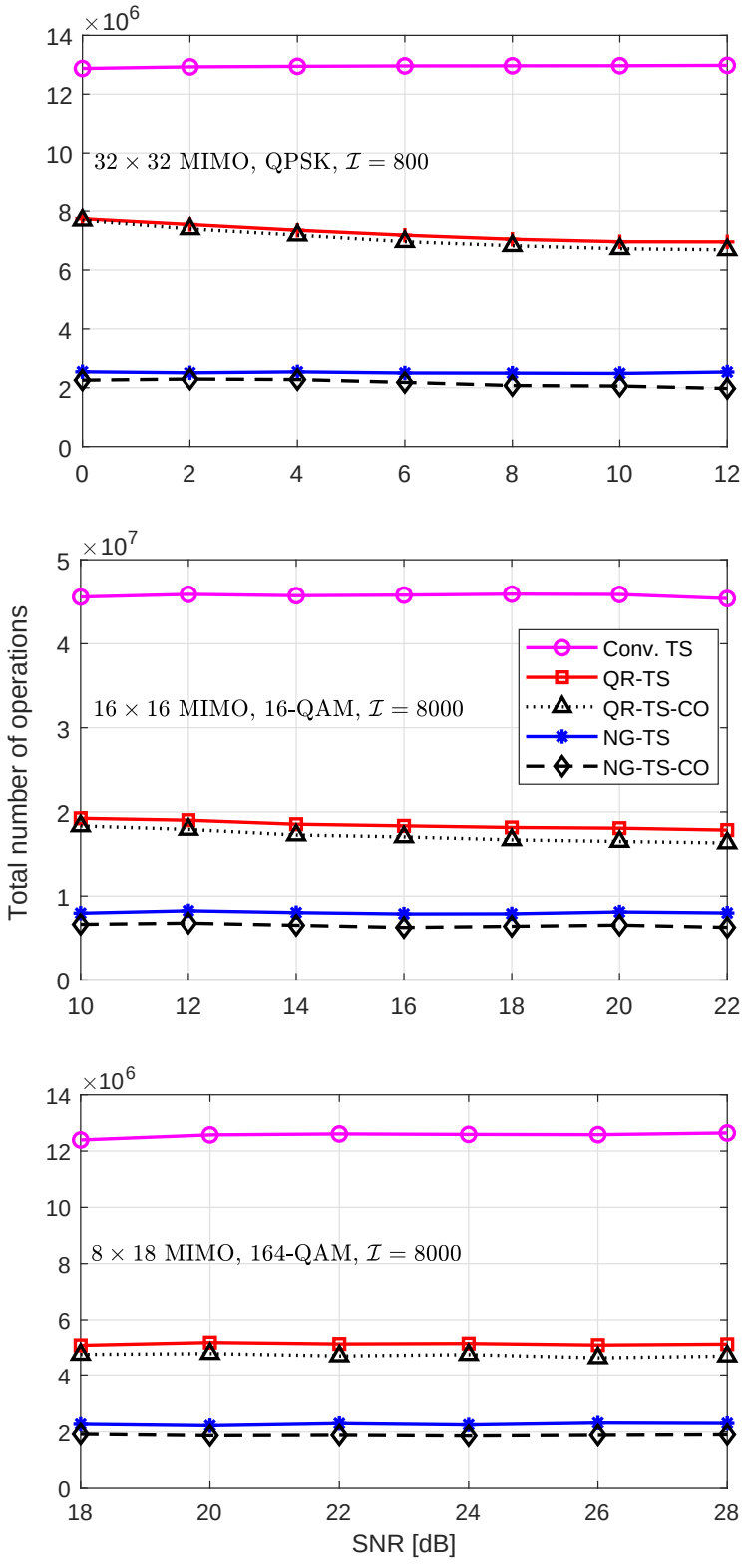


FIG. 3.2. Average computational complexities of the proposed schemes, namely NG-TS and NG-TS-CO, in comparison with those of the conventional TS and QR-TS algorithms. The computational complexity is computed as total number of required multiplications and additions.

3.4. Conclusion

In this chapter, we propose a novel NG-TS algorithm as a solution for the neighbor examination of TS-based symbol detection. The proposed algorithm allows finding the best neighbor with low complexity by using a simplified cost function in a groupwise manner. In addition, a channel ordering scheme is proposed to further optimize the complexity of the proposed NG-TS algorithm. Simulation results show that the proposed NG-TS schemes can achieve up to 85% complexity reduction with respect to the conventional TS algorithm without any performance loss. We note that the proposed algorithm can be applied to other existing TS-based detection algorithms [10, 24–27, 35] as an efficient neighbor examination scheme to reduce their computational complexities.

4. Deep Learning-Aided Tabu Search Detection for Large MIMO Systems

4.1. Motivation and contribution

Although TS detection is considered an efficient symbol-detection algorithm for large MIMO systems [10, 35], it requires many searching iterations to find the optimal solution, causing high computational complexity. The TS algorithm introduced in [27] uses an early termination (ET) criterion to terminate the iterative searching process early after a certain number of iterations when no better solution is found. Although this scheme provides complexity reduction, it can result in significant performance loss because the early terminated searching process does not guarantee the optimal solution. However, the number of searching iterations in the TS algorithm can be reduced with only marginal performance loss if a good initial solution and efficient searching/ET strategies are employed, which can be facilitated by DL. More specifically, we found that the initial solution obtained by a DNN is remarkably more reliable than the conventional linear ZF/MMSE and ordered successive interference cancellation (OSIC) solutions. Furthermore, unlike in the cases of the ZF, MMSE, and OSIC receivers, the initial solution generated by an appropriate activation function in the DNN often has signals very close to or exactly the same as the constellation symbols, even before a quantization is applied. This property can be exploited to efficiently

determine the reliable/unreliable detected symbols in the initial solution. Based on these aspects, the DL-TS algorithm is proposed for complexity reduction of the TS algorithm with ET. Our main contributions are summarized as follows:

- First, we further optimize the DetNet [31, 32] and ScNet [33] architectures to develop the fast-convergence sparsely connected detection network (FS-Net). In this scheme, the network connections are simplified to reduce the complexity. Furthermore, the loss function is improved by taking the correlation between the output of each layer and the desired solution into consideration, resulting in faster convergence. Our simulation results show that the proposed FS-Net architecture achieves improved performance and reduced complexity with respect to DetNet and ScNet. As a result, the FS-Net-based solution is taken as the initial solution of the TS algorithm.

- Secondly, we improve the iterative searching phase of the TS algorithm based on the FS-Net's output. Specifically, by predicting the incorrect symbols in the FS-Net-based initial solution, efficient moves can be made to correct the predicted incorrect symbols. Notably, in the TS algorithm, moving from the current candidate to its best neighbor is equivalent to updating one symbol in the candidate. Therefore, if proper updates are made based on the predicted incorrect symbols, the optimal solution can be reached after fewer searching iterations with a high probability, resulting in complexity reduction of the TS algorithm.

- Furthermore, we propose an adaptive ET criterion incorporated with the FS-Net-based initial solution. We note that in the searching phase of TS algorithm, ET occurs when the number of iterations exceeds a predefined maximum value, specified by a cutoff factor ε [27]. If small ε is set, the final solution can be sub-optimal with a

high probability. By contrast, if ε is set to a large value, search can be performed even after the optimal solution has been found, resulting in many redundant iterations. Motivated by this fact, we propose using an adaptive cutoff factor, which is adjusted based on the accuracy of the FS-Net-based initial solution. As a result, when the initial solution is likely to be accurate, a small number of searching iterations is taken, which leads to a reduction in the overall complexity of the TS algorithm.

The rest of this chapter is organized as follows: Section 4.2 reviews the prior DNN architectures for symbol detection, namely, the FC-DNN, DetNet, and ScNet, followed by the proposal of the FS-Net architecture and the complexity comparison of the considered DNN architectures. Section 4.3 presents the DL-TS detection algorithm. In Section 4.4, the simulation results are shown. Finally, the conclusions are presented in Section 4.5.

4.2. DNNs for MIMO detection

4.2.1. FC-DNN, DetNet, and ScNet architectures

In this section, we review three existing DNNs in the literature for MIMO detection, namely FC-DNN [31], DetNet [31, 32], and ScNet [33], with the focus on complexity analysis, which is our main interest for the design of the proposed FS-Net architecture. For more details on the FC-DNN, DetNet, and ScNet architectures, please refer to [31–33].

1) FC-DNN

In [31] and [32], the performance of the well-known FC-DNN architecture for MIMO detection is examined in two scenarios: fixed and varying channels. It is shown that the

FC-DNN performs well for fixed channels; however, this is an impractical assumption. In contrast, for varying channels, it does not manage to detect the transmitted symbols properly. Therefore, the FC-DNN scheme cannot be employed for symbol detection in practical MIMO systems, and a more sophisticated DNN architecture is required for this purpose.

2) DetNet

In the DetNet, $\hat{\boldsymbol{s}}$ is updated over L layers of the DNN based on mimicking a projected gradient descent-like ML optimization as follows [31, 32]:

$$\begin{aligned}\hat{\boldsymbol{s}}^{[l+1]} &= \Pi\left[\boldsymbol{s} - \delta^{[l]}\frac{\partial\left\|\boldsymbol{y}-\boldsymbol{H}\boldsymbol{s}\right\|^2}{\partial \boldsymbol{s}}\right]_{\boldsymbol{s}=\hat{\boldsymbol{s}}^{[l]}} \\ &= \Pi\left[\hat{\boldsymbol{s}}^{[l]} - \delta^{[l]}\boldsymbol{H}^T\boldsymbol{y} + \delta^{[l]}\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]}\right], \end{aligned} \tag{4.1}$$

where $\Pi[\cdot]$ denotes a nonlinear projection operator and $\delta^{[l]}$ is a step size. The operation and architecture of the lth layer of the DetNet is illustrated in Fig. 4.1. It is shown that $\hat{\boldsymbol{s}}^{[l]}$ and $\boldsymbol{v}^{[l]}$, which are not only the output of the lth layer but also the input of the $(l+1)$th layer, are updated as follows:

$$\boldsymbol{q}^{[l]} = \hat{\boldsymbol{s}}^{[l-1]} - \delta_1^{[l]}\boldsymbol{H}^T\boldsymbol{y} + \delta_2^{[l]}\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l-1]}, \tag{4.2}$$

$$\boldsymbol{x}^{[l]} = \left[\boldsymbol{v}^{[l-1]}, \boldsymbol{q}^{[l]}\right]^T, \tag{4.3}$$

$$\boldsymbol{z}^{[l]} = \sigma\left(\boldsymbol{W}_1^{[l]}\boldsymbol{x}^{[l]} + \boldsymbol{b}_1^{[l]}\right), \tag{4.4}$$

$$\hat{\boldsymbol{s}}^{[l]} = \psi_t\left(\boldsymbol{W}_2^{[l]}\boldsymbol{z}^{[l]} + \boldsymbol{b}_2^{[l]}\right), \tag{4.5}$$

$$\boldsymbol{v}^{[l]} = \boldsymbol{W}_3^{[l]}\boldsymbol{z}^{[l]} + \boldsymbol{b}_3^{[l]}, \tag{4.6}$$

where $\hat{\boldsymbol{s}}^{[0]} = \boldsymbol{v}^{[0]} = \mathbf{0}$, with $\mathbf{0}$ being an all-zero vector of an appropriate size. In (4.3), $\boldsymbol{q}^{[l]}$ and $\boldsymbol{v}^{[l-1]}$ are concatenated into a single input vector $\boldsymbol{x}^{[l]}$, and in (4.2)–(4.6),

$$\left\{ \boldsymbol{W}_1^{[l]}, \boldsymbol{W}_2^{[l]}, \boldsymbol{W}_3^{[l]}, \boldsymbol{b}_1^{[l]}, \boldsymbol{b}_2^{[l]}, \boldsymbol{b}_3^{[l]}, \delta_1^{[l]}, \delta_2^{[l]} \right\}$$

are the training parameters in the lth layer of the DetNet. In (4.4), $\sigma(\cdot)$ represents the rectified linear unit (ReLU) activation function. Furthermore, $\psi_t(\cdot)$ in (4.5), defined as $\psi_t(x) = -q + \frac{1}{|t|}\sum_{i\in\Omega}[\sigma(x+i+t) - \sigma(x+i-t)]$ with $q = 1, \Omega = \{0\}$ for QPSK and $q = 3, \Omega = \{-2, 0, 2\}$ for 16-QAM, guarantees that the amplitudes of the elements of $\hat{\boldsymbol{s}}^{[l]}$ are in the range $[-1, 1]$ for QPSK and $[-3, 3]$ for 16-QAM, as illustrated in Fig. 4.2. The final detected symbol vector is given as $\hat{\boldsymbol{s}} = \mathcal{Q}\left(\hat{\boldsymbol{s}}^{[L]}\right)$, where $\mathcal{Q}(\cdot)$ quantizes each element of $\hat{\boldsymbol{s}}^{[L]}$ to its closest real-constellation symbol in $\mathcal{A}$.

It is observed from (4.4)–(4.6) and Fig. 4.1 that an additional input vector $\boldsymbol{v}$ and the full connections between the input and output vectors in every layer makes the network architecture of the DetNet complicated. This not only causes its high computational complexity, as will be shown in Section 4.2.3, but also makes it difficult to optimize, resulting in its relatively low performance, as will be shown in Section 4.4. Therefore, the ScNet was introduced in [33] for complexity reduction and performance improvement.

3) ScNet

The ScNet also follows the update process in (4.1), but it simplifies the DetNet architecture by removing $\boldsymbol{v}$ [33]. As a result, $\{\boldsymbol{W}_3, \boldsymbol{b}_3\}$ is also removed. Furthermore, it is observed in (4.1) that the first element of $\hat{\boldsymbol{s}}^{[l+1]}$ only depends on the first element of $\boldsymbol{q}^{[l]}$, which implies that the full connection between all elements of $\boldsymbol{q}^{[l]}$ and $\hat{\boldsymbol{s}}^{[l+1]}$ is unnecessary. As a result, the input and output of each layer of the ScNet are directly connected in the

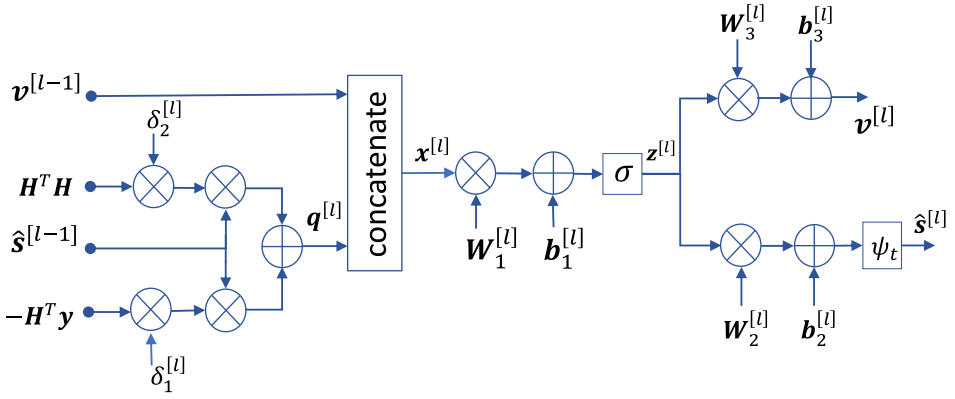


FIG. 4.1. The lth layer of the DetNet architecture

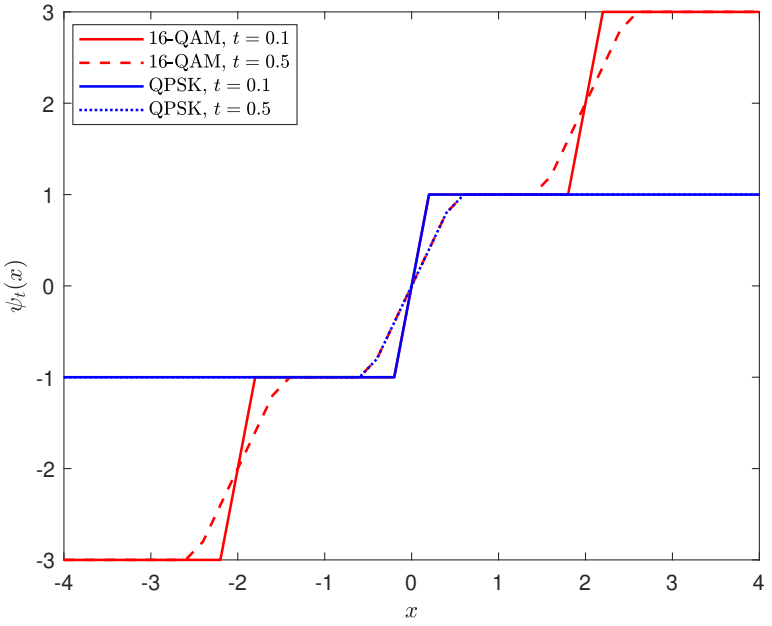


FIG. 4.2. $\psi_t(x)$ in DetNet, ScNet, and FS-Net.

element-wise manner. Consequently, $\left\{ \boldsymbol{W}_2^{[l]}, \boldsymbol{b}_2^{[l]} \right\}$ is removed, and the weight matrix $\boldsymbol{W}_1^{[l]}$ is reduced to a weight vector $\boldsymbol{w}^{[l]}$ of size $3N \times 1$.

The operation and architecture of the ScNet is illustrated in Fig. 4.3. Similar to DetNet, ScNet is initialized with $\hat{\boldsymbol{s}}^{[0]} = \boldsymbol{v}^{[0]} = \boldsymbol{0}$. Then, the output of the lth layer, i.e., $\hat{\boldsymbol{s}}^{[l]}$, is

updated as follows:

$$\boldsymbol{x}^{[l]} = [\boldsymbol{H}^T\boldsymbol{y}, \boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l-1]}, \hat{\boldsymbol{s}}^{[l-1]}]^T, \tag{4.7}$$

$$\hat{\boldsymbol{s}}^{[l]} = \psi_t(\boldsymbol{w}^{[l]} \odot \boldsymbol{x}^{[l]} + \boldsymbol{b}_l), \tag{4.8}$$

where $\boldsymbol{w}^{[l]} \odot \boldsymbol{x}^{[l]}$ denotes the element-wise multiplication of $\boldsymbol{w}^{[l]}$ and $\boldsymbol{x}^{[l]}$.

The simulation results in [33] show that for BER $= 10^{-4}$, the ScNet achieves an approximate SNR gain of 1 dB over the DetNet with lower complexity. However, one drawback of the ScNet is that the input vector $\boldsymbol{x}$ has the size of $(3N \times 1)$, which is three times larger than that of $\boldsymbol{q}$ in the DetNet for containing the information for $\boldsymbol{y}$, $\boldsymbol{H}$, and $\hat{\boldsymbol{s}}^{[l-1]}$. This may result in unnecessary computational complexity of the ScNet. Furthermore, the loss function of the DetNet and ScNet does not guarantee fast convergence. These observations on the input vector and the loss function of the DetNet and ScNet motivate us to propose the FS-Net for complexity reduction and performance improvement in the next section.

4.2.2. Proposed FS-Net architecture

1) Network architecture

Inheriting the DetNet and ScNet, the proposed FS-Net is also motivated by the updating process in (4.1). We note that (4.1) can be rewritten as

$$\hat{\boldsymbol{s}}^{[l+1]} = \Pi\left[\hat{\boldsymbol{s}}^{[l]} + \delta^{[l]}(\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]} - \boldsymbol{H}^T\boldsymbol{y})\right], \tag{4.9}$$

which shows that the contributions of $\hat{\boldsymbol{s}}^{[l]}$ and $\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]} - \boldsymbol{H}^T\boldsymbol{y}$ to $\hat{\boldsymbol{s}}^{[l+1]}$ are different. Therefore, their elements should be processed by different weights and biases. Furthermore,

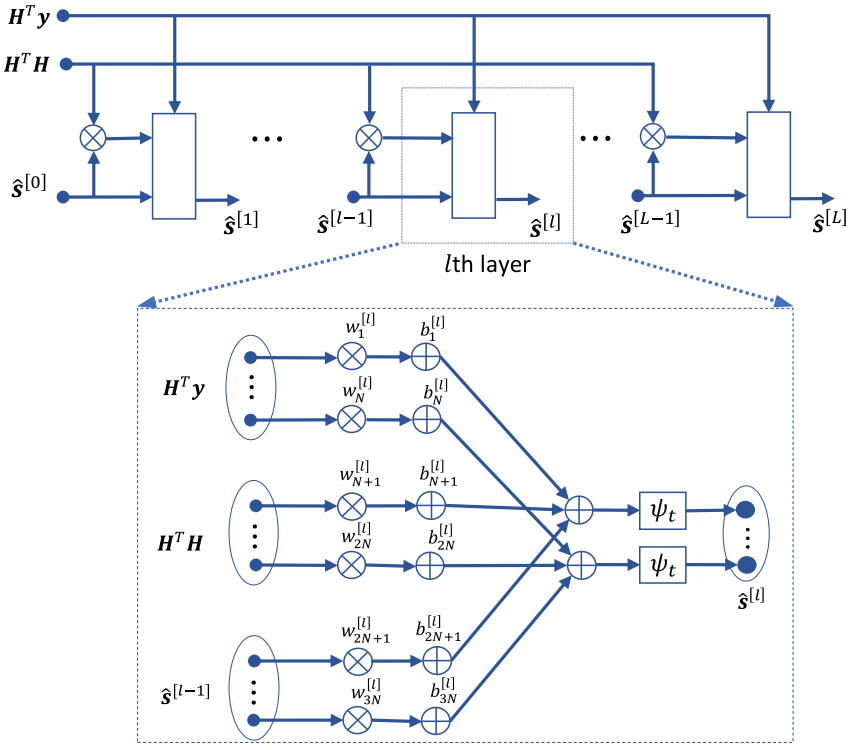


FIG. 4.3. The ScNet architecture

(4.9) also implies that the elements at the same position of $\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]}$ and $\boldsymbol{H}^T\boldsymbol{y}$ can be multiplied by the same weight. Therefore, in the proposed FS-Net, we set the input vector of the $(l+1)$th layer to

$$\boldsymbol{x}^{[l]} = \left[\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]} - \boldsymbol{H}^T\boldsymbol{y}\right]^T \in \mathbb{R}^{2N\times 1},$$

whose size is only 2/3 that of $\boldsymbol{x}^{[l]}$ in (4.7) for the ScNet. Furthermore, the FS-Net follows the sparse connection of ScNet. Consequently, in each layer of the FS-Net, there are only $2N$ element-wise connections between the input and output, whereas the ScNet has $3N$. The operation and architecture of the proposed FS-Net network is illustrated in Fig. 4.4. Furthermore, Algorithm 4 summarizes the FS-Net scheme for MIMO detection. The output of each layer is updated in step 4, where $\boldsymbol{x}^{[l]}$ is obtained in step 3.

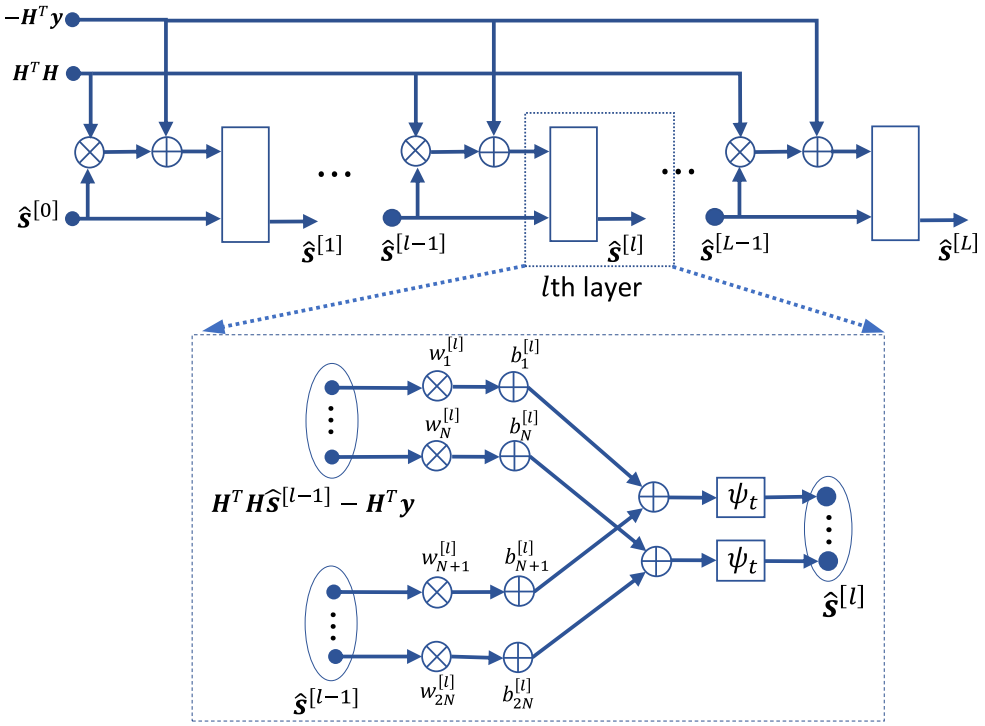


FIG. 4.4. The FS-Net architecture

2) Loss function

The loss function of the DetNet and ScNet is given as [31,32]

$$\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \sum_{l=1}^{L} \log(l) \left\| \boldsymbol{s} - \hat{\boldsymbol{s}}^{[l]} \right\|^2, \tag{4.10}$$

which measures the total weighted distance between the transmitted vector $\boldsymbol{s}$ and the outputs of all the layers, i.e., $\hat{\boldsymbol{s}}^{[l]}, l = 1, \ldots, L$. The DetNet and ScNet are trained to optimize the parameter set $\left\{ \boldsymbol{W}_i^{[l]}, \boldsymbol{b}_i^{[l]} \right\}$, $i = 1, 2, 3$, $l = 1, \ldots, L$, such that $\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{s}})$ is minimized. As a result, $\hat{\boldsymbol{s}}$ can converge to $\boldsymbol{s}$. However, the disadvantage of this mechanism is that the moving direction of $\hat{\boldsymbol{s}}^{[l]}$ cannot be controlled, and the moves can be made along an inefficient path with slow convergence.

In the proposed FS-Net, we consider the correlation between $\hat{\boldsymbol{s}}^{[l]}$ and $\boldsymbol{s}$ in the loss

function for better training the FS-Net. Specifically, the loss function of the FS-Net is redefined as

$$\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \sum_{l=1}^{L} \log(l) \left[\left\| \boldsymbol{s} - \hat{\boldsymbol{s}}^{[l]} \right\|^2 + \beta r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s}) \right], \tag{4.11}$$

where $r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s}) = 1 - \frac{\left|\boldsymbol{s}^T \hat{\boldsymbol{s}}^{[l]}\right|}{\|\boldsymbol{s}\| \|\hat{\boldsymbol{s}}^{[l]}\|}$. Based on the Cauchy–Schwarz inequality, we have $|\boldsymbol{s}^T \hat{\boldsymbol{s}}^{[l]}| \leq \|\boldsymbol{s}\| \, \|\hat{\boldsymbol{s}}^{[l]}\|$, where equality occurs if $\boldsymbol{s} = c\hat{\boldsymbol{s}}^{[l]}$ with a constant c.

The DetNet, ScNet, and FS-Net schemes initialize $\hat{\boldsymbol{s}}^{[0]}$ as $\mathbf{0}$ and update $\hat{\boldsymbol{s}}^{[l]}$ over L layers to approximate $\boldsymbol{s}$. This can be considered as sequential moves starting from the coordinate $\mathbf{0}$ over $\hat{\boldsymbol{s}}^{[1]}, \ldots, \hat{\boldsymbol{s}}^{[L-1]}$ to reach $\hat{\boldsymbol{s}}^{[L]} \approx \boldsymbol{s}$. By minimizing the loss function in (4.11), we have $r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s}) \to 0$, or equivalently, $\boldsymbol{s} \approx \hat{\boldsymbol{s}}^{[l]}, l = 1, 2, \ldots, L$, which enables the moves to be in a specific direction, which is the position of $\boldsymbol{s}$ in a hypersphere. This can significantly shorten the path of the moves to reach $\boldsymbol{s}$, which results in a reduced number of required layers in the FS-Net. In (4.11), β is used to adjust the contribution of $r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s})$ to the loss function, which is optimized by simulations. We note that $r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s})$ should have a smaller contribution than $\|\boldsymbol{s} - \hat{\boldsymbol{s}}^{[l]}\|^2$ in the loss value because its role is to improve the convergence speed, whereas $\|\boldsymbol{s} - \hat{\boldsymbol{s}}^{[l]}\|^2$ is the primary factor deciding the approximation accuracy. Therefore, β should be selected such that $0 < \beta < 1$. However, a very small β does not guarantee good convergence of $\hat{\boldsymbol{s}}^{[l]}$ to $\boldsymbol{s}$, while our simulations show that a large β worsens the performance of the FS-Net. Therefore, we chose $\beta = 0.5$ as the base value. Our simulation results in Section 4.4 show that the proposed FS-Net achieves not only complexity reduction, but also performance improvement with respect to the conventional DetNet and ScNet schemes.

Algorithm 4 FS-Net scheme for MIMO detection

Input: $\boldsymbol{H}, \boldsymbol{y}$.
Output: $\hat{\boldsymbol{s}}$.
1: $\hat{\boldsymbol{s}}^{[0]} = \mathbf{0}$
2: **for** $l = 1 \rightarrow L$ **do**
3: $\quad \boldsymbol{x}^{[l]} = \left[\hat{\boldsymbol{s}}^{[l-1]}, \boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l-1]} - \boldsymbol{H}^T\boldsymbol{y}\right]^T$
4: $\quad \hat{\boldsymbol{s}}^{[l]} = \psi_t(\boldsymbol{w}^{[l]} \odot \boldsymbol{x}^{[l]} + \boldsymbol{b})$
5: **end for**
6: $\hat{\boldsymbol{s}} = \mathcal{Q}\left(\hat{\boldsymbol{s}}^{[L]}\right)$

TABLE 4.1. Computational complexity comparison of the DetNet, ScNet, and FS-Net architectures

DNN architectures	Total complexity	Layer complexity
DetNet	$\mathcal{C}_{\text{DetNet}}^{\text{QPSK}} = (18L + 2M - 1)N^2 + (2M - 1 + L)N$	$18N^2 + N$
	$\mathcal{C}_{\text{DetNet}}^{\text{16-QAM}} = (50L + 2M - 1)N^2 + (2M - 1 + L)N$	$50N^2 + N$
ScNet	$\mathcal{C}_{\text{ScNet}} = (2L + 2M - 1)N^2 + (2M - 1 + 5L)N$	$2N^2 + 5N$
FS-Net	$\mathcal{C}_{\text{FS-Net}} = (2L + 2M - 1)N^2 + (2M - 1 + 4L)N$	$2N^2 + 4N$

4.2.3. Complexity comparison of DetNet, ScNet, and FS-Net

1) Computational complexity of the DetNet

We note that the computations of $\boldsymbol{H}^T\boldsymbol{y}$ and $\boldsymbol{H}^T\boldsymbol{H}$ require $N(2M-1)$ and $N^2(2M-1)$ operations, respectively, and are performed once in the first layer. The complexities required in (4.2), (4.4)–(4.6) depend on the modulation scheme.

For QPSK, the sizes of $\boldsymbol{v}^{[l]}$, $l = 1, \ldots, L$, is set to $N \times 1$ [31], leading to $\boldsymbol{x}^{[l]} \in \mathbb{R}^{2N\times 1}$. As a result, the sizes of $\boldsymbol{W}_1^{[l]}$, $\boldsymbol{W}_3^{[l]}$, $\boldsymbol{b}_1^{[l]}$, and $\boldsymbol{b}_3^{[l]}$ can be inferred from the size of $\boldsymbol{z}^{[l]}$, which is set to $2N \times 1$ [31], as follows: $\boldsymbol{W}_1^{[l]} \in \mathbb{R}^{2N\times 2N}$, $\boldsymbol{b}_1^{[l]} \in \mathbb{R}^{2N\times 1}$, $\boldsymbol{W}_3^{[l]} \in \mathbb{R}^{N\times 2N}$, and $\boldsymbol{b}_3^{[l]} \in \mathbb{R}^{N\times 1}$. Furthermore, because $\hat{\boldsymbol{s}}^{[l+1]} \in \mathbb{R}^{N\times 1}$, we have $\boldsymbol{W}_2^{[l]} \in \mathbb{R}^{N\times 2N}$, and $\boldsymbol{b}_2^{[l]} \in \mathbb{R}^{N\times 1}$. Then, given $\boldsymbol{H}^T\boldsymbol{y}$ and $\boldsymbol{H}^T\boldsymbol{H}$, the total complexity required in each layer is $18N^2 + N$ operations, including $2N^2 + N$, $8N^2$, $4N^2$, and $4N^2$ operations for (4.2),

(4.4)–(4.6), respectively. Therefore, the total complexity of all L layers of the DetNet is given as

$$\begin{aligned}\mathcal{C}_{\text{DetNet}}^{\text{QPSK}} &= N(2M-1) + N^2(2M-1) + L(18N^2+N) \\ &= (18L+2M-1)N^2 + (2M-1+L)N. \end{aligned} \tag{4.12}$$

For 16-QAM, $\boldsymbol{v}^{[l]}$ and $\boldsymbol{z}^{[l]}$ are set to $\boldsymbol{v}^{[l]} \in \mathbb{R}^{2N\times 1}$ and $\boldsymbol{z}^{[l]} \in \mathbb{R}^{4N\times 1}$ [31]. Therefore, we have $\boldsymbol{x}^{[l]} \in \mathbb{R}^{3N\times 1}$, resulting in $\boldsymbol{W}_1^{[l]} \in \mathbb{R}^{4N\times 3N}$, $\boldsymbol{b}_1^{[l]} \in \mathbb{R}^{4N\times 1}$, $\boldsymbol{W}_2^{[l]} \in \mathbb{R}^{N\times 4N}$, $\boldsymbol{b}_2^{[l]} \in \mathbb{R}^{N\times 1}$, $\boldsymbol{W}_3^{[l]} \in \mathbb{R}^{2N\times 4N}$, and $\boldsymbol{b}_3^{[l]} \in \mathbb{R}^{2N\times 1}$. Then, given $\boldsymbol{H}^T\boldsymbol{y}$ and $\boldsymbol{H}^T\boldsymbol{H}$, the complexities required in (4.2) and (4.4)–(4.6) are $2N^2+N$, $24N^2$, $8N^2$, and $16N^2$ operations, respectively. As a result, the total complexity required in each layer of the DetNet with 16-QAM is $50N^2+N$ operations. Therefore, the total complexity of all L layers of the DetNet is given as

$$\begin{aligned}\mathcal{C}_{\text{DetNet}}^{\text{16-QAM}} &= N(2M-1) + N^2(2M-1) + L(50N^2+N) \\ &= (50L+2M-1)N^2 + (2M-1+L)N. \end{aligned} \tag{4.13}$$

2) Computational complexity of the ScNet

Given $\boldsymbol{H}^T\boldsymbol{H}$, the computation of $\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]}$ in (4.7) requires $2N^2-N$ operations. Furthermore, because $\boldsymbol{x}^{[l]} \in \mathbb{R}^{3N\times 1}$, we have $\boldsymbol{w}^{[l]}, \boldsymbol{b}^{[l]} \in \mathbb{R}^{3N\times 1}$, and the computation in (4.8) requires only $6N$ operations. Consequently, the complexity of each layer of the ScNet architecture is $2N^2+5N$. Taking the complexities of computing $\boldsymbol{H}^T\boldsymbol{y}$ and $\boldsymbol{H}^T\boldsymbol{H}$ into

consideration, the ScNet architecture requires

$$\begin{aligned}\mathcal{C}_{\text{ScNet}} &= N(2M-1) + N^2(2M-1) + L(2N^2+5N) \\ &= (2M-1+2L)N^2 + (2M-1+5L)N \end{aligned} \tag{4.14}$$

operations in total.

3) Computational complexity of the FS-Net

The output of each layer is updated in step 4, where $\boldsymbol{x}^{[l]}$ is obtained in step 3 of Algorithm 4 with the requirement of $2N^2$ operations to compute $\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{s}}^{[l]} - \boldsymbol{H}^T\boldsymbol{y}$ when $\boldsymbol{H}^T\boldsymbol{y}$ and $\boldsymbol{H}^T\boldsymbol{H}$ are given. In the FS-Net architecture, we have $\boldsymbol{w}^{[l]}, \boldsymbol{b}^{[l]} \in \mathbb{R}^{2N\times 1}$. Therefore, the computation in each layer requires only $2N^2+4N$ operations. We recall that the complexities of computing $\boldsymbol{H}^T\boldsymbol{y}$ and $\boldsymbol{H}^T\boldsymbol{H}$ are $N(2M-1)$ and $N^2(2M-1)$ operations. As a result, the complexity of the entire FS-Net is given as

$$\begin{aligned}\mathcal{C}_{\text{FS-Net}} &= N(2M-1) + N^2(2M-1) + L(2N^2+4N) \\ &= (2M-1+2L)N^2 + (2M-1+4L)N. \end{aligned} \tag{4.15}$$

The computational complexities of the DetNet, ScNet, and FS-Net are summarized in Table 4.1. In this table, we also present the complexity of each layer of the considered architectures. In Table 4.1, the proposed FS-Net requires approximately $16LN^2$ fewer operations than the DetNet for QPSK, and that number for 16-QAM is $48LN^2$. Compared to the ScNet architecture, the complexity reduction of the FS-Net is LN operations. It is worth noting that the proposed FS-Net not only achieves complexity reduction but also

significantly improves the BER performance with respect to the conventional DetNet and ScNet, as will be shown in Section 4.4. This implies that to achieve the same performance, the FS-Net requires a shallower neural network than the DetNet and ScNet. As a result, a considerable reduction in the training time and complexity can be attained by the FS-Net compared to the DetNet and ScNet.

4.3. Deep Learning-Aided Tabu Search detection

4.3.1. Problem formulation

The TS algorithm starts with an initial candidate and sequentially moves over $\mathcal{I}_{\mathrm{UB}}$ candidates for $\mathcal{I}_{\mathrm{UB}}$ iterations. In each iteration, all the non-tabu neighbors of the current candidate $\boldsymbol{c}$ are examined to find the best neighbor $\boldsymbol{x}^{\star}$ with the smallest ML metric, i.e.,

$$\boldsymbol{x}^{\star} = \arg \min_{\boldsymbol{x} \in \mathcal{N}(\boldsymbol{c})} \phi(\boldsymbol{x}), \tag{4.16}$$

where $\mathcal{N}(\boldsymbol{c})$ consists of non-tabu neighboring vectors inside alphabet $\mathcal{A}^N$ with the smallest distance to $\boldsymbol{c}$, i.e.,

$$\mathcal{N}(\boldsymbol{c}) = \left\{ \boldsymbol{x} \in \mathcal{A}^N \backslash \mathcal{L}, |\boldsymbol{x} - \boldsymbol{c}| = \theta_{min} \right\}, \tag{4.17}$$

where $\mathcal{A}^N \backslash \mathcal{L}$ denotes the alphabet $\mathcal{A}^N$ excluding the tabu vectors kept in the tabu list $\mathcal{L}$, and θ_{min} is the minimum distance between two constellation points in a plane. Furthermore, the ML metric $\phi(\boldsymbol{x})$ can be expressed as [35]

$$\phi(\boldsymbol{x}) = \left\| \boldsymbol{u} + \boldsymbol{h}_d \delta_d \right\|^2, \tag{4.18}$$

where $\boldsymbol{u} = \boldsymbol{y} - \boldsymbol{H}\,\boldsymbol{c}$, δ_d is the single nonzero element of $\boldsymbol{c} - \boldsymbol{x} = [0, \ldots, 0, \delta_d, 0, \ldots, 0]^T$ and $\boldsymbol{h}_d$ is the dth column of $\boldsymbol{H}$. In this study, we refer to d as the *difference position* of a neighbor, in which the candidate and its neighbor are different. For example, if the current candidate is $\boldsymbol{c} = [1, -3, 1, 3]^T$, then $d = 3$ is the difference position for $\boldsymbol{x} = [1, -3, -1, 3]^T$ because $\boldsymbol{c}$ and $\boldsymbol{x}$ are only different with respect for the third element.

After the best neighbor is determined with (4.16), it becomes the candidate in the next iteration, and the determination of the best neighbor of a new candidate is performed. By this iterative manner, the final solution $\hat{\boldsymbol{s}}_{TS}$ is determined as the best candidate visited so far, i.e., $\hat{\boldsymbol{s}}_{TS} = \arg\min_{\boldsymbol{c}\in\mathcal{V}} \{\phi(\boldsymbol{c})\}$, where $\mathcal{V}$ is the set of all visited candidates over $\mathcal{I}_{\mathrm{UB}}$ searching iterations.

The computational complexity of TS algorithms is proportional to the number of searching iterations, i.e., $\mathcal{I}_{\mathrm{UB}}$. In large MIMO systems, the number of neighbors in each iteration and the dimension of the neighboring vectors are large. Therefore, the complexity to find the best neighbor in each iteration becomes high in large MIMO systems. Furthermore, an extremely large $\mathcal{I}_{\mathrm{UB}}$ is required to guarantee that the near-optimal solution is found. Consequently, the complexity of the TS algorithm can be excessively high in large MIMO systems. To reduce the complexity of TS in large MIMO systems, an ET scheme can be employed. Specifically, a cutoff factor ε, $0 < \varepsilon < 1$, is used to terminate the iterative searching process early after $\mathcal{I}_{\mathrm{e}} = \varepsilon\mathcal{I}_{\mathrm{UB}}$ iterations in which no better solution is found. As a result, the number of searching iterations in the TS algorithm with ET is

$$\mathcal{I} = \min\{\mathcal{I}_0 + \mathcal{I}_{\mathrm{e}}, \mathcal{I}_{\mathrm{UB}}\}, \tag{4.19}$$

where $\mathcal{I}_0$ is the number of iterations after which no better solution is found. The TS algorithm without ET requires $\mathcal{I}_{\mathrm{UB}}$ searching iterations, which is also the upper bound on

the number of iterations required when ET is applied. From (4.19), the following remarks are noted.

Remark 4.1. If ET occurs, the best solution found after $\mathcal{I}_0$ iterations and that found after $\mathcal{I} = \mathcal{I}_0 + \mathcal{I}_\mathrm{e}$ iterations are the same, which is the final solution of the TS algorithm, i.e., $\hat{\boldsymbol{s}}_{TS}$. Therefore, the earlier $\hat{\boldsymbol{s}}_{TS}$ is found, the smaller $\mathcal{I}_0$ is required. This objective can be achieved by starting the moves in the TS algorithm with a good initial solution. Furthermore, if the initial solution is well taken such that it is likely to be a near-optimal solution, then no further searching iteration is required. In this case, the complexity of the TS algorithm becomes only that involved in finding the initial solution.

Remark 4.2. Another approach to find the optimal solution $\hat{\boldsymbol{s}}_{TS}$ earlier and reduce $\mathcal{I}_0$ is to make efficient moves during the searching iterations in the TS algorithm. In other words, the moves can be guided so that $\hat{\boldsymbol{s}}_{TS}$ is reached earlier.

Remark 4.3. With the conventional ET criterion, $\mathcal{I}_e$ is fixed to $\mathcal{I}_e = \varepsilon \mathcal{I}_\mathrm{UB}$. In large MIMO systems, a large $\mathcal{I}_\mathrm{UB}$ is required to guarantee that the optimal solution is found, which results in a large $\mathcal{I}_e$. However, we note that once ET occurs, the further search over $\mathcal{I}_e$ iterations do not result in any performance improvement while causing significant computational burden for the TS algorithm. For example, in a 32×32 MIMO system with QPSK, $\mathcal{I}_\mathrm{UB} = 800$ should be used to approximately achieve the performance of the SD scheme [35]. For $\varepsilon = 0.25$, $\mathcal{I}_e = 0.25 \times 800 = 200$ iterations are required before the termination, whereas $\hat{\boldsymbol{s}}_{TS}$ was already found in the $(\mathcal{I}_0 = \mathcal{I} - 200)$th iteration. Therefore, a more efficient ET criterion is required to reduce the complexity of the TS algorithm with ET.

Remarks 4.1–4.3 motivate us to propose a TS-based detection algorithm for complexity reduction with three design objectives: taking a good initial solution, using efficient moves

in searching iterations so that $\hat{\boldsymbol{s}}_{TS}$ is reached as soon as possible, and terminating the TS algorithm early based on an efficient ET criterion. In the next subsection, we propose the DL-TS algorithm with the application of DL to the TS detection for those objectives.

4.3.2. Proposed DL-TS algorithm

The main ideas of the proposed DL-TS algorithm can be explained as follows.

1) DL-aided initial solution

Unlike the conventional TS algorithms, in which the ZF, MMSE, or OSIC solution is taken as the initial solution [27], the DL-TS algorithm employs a DNN to generate the initial solution.

In this scheme, the most important task is to choose a DNN architecture that is not only able to approximate the transmitted signal vector with high accuracy, but also has low computational complexity. The FC-DNN architectures is considered in most of the prior studies on the application of DL to the symbol detection in various scenarios, such as molecular communication [29] and OFDM systems [30]. This architecture can require an excessively high computational complexity because of its dense connections in layers, especially in large MIMO systems, where the sizes of input and output vectors are large. Furthermore, it is shown in [31] and [32] that the FC-DNN cannot manage to detect the symbols properly for varying channels in large MIMO systems. Therefore, the FC-DNNs are not considered in our study, where the DNN architecture needs to find the initial solution for the TS algorithm with optimal performance-complexity tradeoff.

Among the DetNet, ScNet, and FS-Net, the FS-Net requires the lowest complexity while achieving the best performance, which is demonstrated in Section 4.4. Therefore,

we propose employing the FS-Net to find the initial solution in the DL-TS algorithm. Specifically, $\hat{\boldsymbol{s}}$ obtained in step 6 of Algorithm 4 is taken as the initial solution of the DL-TS algorithm. As a result, the required complexity for this initialization phase is given in (4.15).

2) Efficient moves in searching iterations

In the TS algorithm with ET, the complexity can be reduced if efficient moves are made during the iterative searching process, so that $\hat{\boldsymbol{s}}_{TS}$ is found earlier, as discussed in Remark 4.2. For this purpose, we propose exploiting the difference between $\hat{\boldsymbol{s}}^{[L]}$ and $\hat{\boldsymbol{s}}$, which are the output of the last layer and the final solution of the FS-Net, respectively. We recall that $\hat{\boldsymbol{s}}^{[L]} \in \mathbb{R}^{N\times 1}$ can contain elements both inside and outside the alphabet $\mathcal{A}$, as observed from step 4 in Algorithm 4 and Fig. 4.2. By contrast, we have $\hat{\boldsymbol{s}} = \mathcal{Q}\left(\hat{\boldsymbol{s}}^{[L]}\right) \in \mathcal{A}^N$.

Let $\boldsymbol{e}$ denote the distance between the elements of $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$, i.e., $\boldsymbol{e} = \left|\hat{\boldsymbol{s}}^{[L]} - \hat{\boldsymbol{s}}\right| = [e_1, e_2, \ldots, e_N]^T$. For QAM signals, the distance between two neighboring real symbols is two. Furthermore, from Fig. 4.2, we have $\hat{s}^{[L]} \in [-1, 1]$ for QPSK and $\hat{s}^{[L]} \in [-3, 3]$ for 16-QAM. Therefore, we have $0 \leq e_n \leq 1, n = 1, 2, \ldots, N$. It is observed that if $e_n \approx 0$, there is a high probability that the nth symbol in $\hat{\boldsymbol{s}}$ is correctly approximated by the FS-Net, i.e., $s_n = \hat{s}_n$. By contrast, if $e_n \approx 1$, there is a high probability that $\hat{s}_n$ is an erroneous estimate, i.e., $s_n \neq \hat{s}_n$. Therefore, by examining the elements of $\boldsymbol{e}$, we can determine the elements of $\hat{\boldsymbol{s}}$ with high probabilities of errors.

Example 1: Consider a MIMO system with $N = 2N_t = 8$, QPSK, and $\hat{\boldsymbol{s}}^{[L]} = [0.1, -0.9, -0.2, 1, 0.25, 0.9, -1, 1]^T$, $\hat{\boldsymbol{s}} = [1, -1, -1, 1, 1, 1, -1, 1]^T$. Then, we have $\boldsymbol{e} = \left|\hat{\boldsymbol{s}}^{[L]} - \hat{\boldsymbol{s}}\right| = [0.9, 0.1, 0.8, 0, 0.75, 0.1, 0, 0]^T$, which implies that $\hat{s}_1, \hat{s}_3$ and $\hat{s}_5$ can be incorrect with high probabilities.

In the following analysis, we refer to the symbols of $\hat{\boldsymbol{s}}$ with high probabilities of being

incorrect as the *predicted incorrect symbols*. Furthermore, the nth element in $\hat{\boldsymbol{s}}$ is determined as a predicted incorrect symbol if e_n is beyond a predefined error-threshold γ, i.e., if $e_n > \gamma$. The error-threshold should be inversely proportional to the SNR. Furthermore, because $0 \leq e_n \leq 1, n = 1, 2, \ldots, N$, we set γ to

$$\gamma = \min\left\{\frac{1}{\mathrm{SNR}}, 0.5\right\}. \tag{4.20}$$

Let n_e denote the number of predicted incorrect symbols in $\hat{\boldsymbol{s}}$ ($n_e = 3$ in *Example* 1), and let $\mathbb{S}$ be the set of all candidates obtained by correcting $\hat{\boldsymbol{s}}$. Then we have $\mathbb{S} = \mathcal{S}^{(1)} \cup \mathcal{S}^{(2)} \cup \ldots \cup \mathcal{S}^{(n_e)}$, where $\mathcal{S}^{(k)}, 1 \leq k \leq n_e$, is a subset of $\mathbb{S}$ obtained by correcting k elements of $\hat{\boldsymbol{s}}$. The number of candidates in $\mathcal{S}^{(k)}$ is given as $\left|\mathcal{S}^{(k)}\right| = C_k^{n_e} \times (Q-1)^k = \frac{n_e!(Q-1)^k}{k!(n_e-k)!}$, where $C_k^{n_e} = \frac{n_e!}{k!(n_e-k)!}$ is the number of combinations for choosing k out of n_e predicted incorrect symbols, and Q is the number of real symbols in $\mathcal{A}$ with $Q = \{2, 4, 8\}$ for QPSK, 16-QAM, and 64-QAM, respectively. Consequently, the total number of possible corrected vectors in $\mathbb{S}$ is $|\mathbb{S}| = \sum_{k=1}^{n_e} \frac{n_e!(Q-1)^k}{k!(n_e-k)!}$. Now, denoting $\bar{\boldsymbol{s}}^\star$ as the best vector obtained by correcting $\hat{\boldsymbol{s}}$, we have

$$\bar{\boldsymbol{s}}^\star = \arg\min_{\bar{\boldsymbol{s}} \in \mathbb{S}} \|\boldsymbol{y} - \boldsymbol{H}\bar{\boldsymbol{s}}\|^2. \tag{4.21}$$

In the case of large n_e and high-order QAM schemes, $|\mathbb{S}|$ becomes large; hence, high complexity is required to find $\bar{\boldsymbol{s}}^\star$ in (4.21).

However, we note that correcting a symbol in $\hat{\boldsymbol{s}}$ is equivalent to a move from $\hat{\boldsymbol{s}}$ to one of its neighboring vectors in an iteration of the TS algorithm. Specifically, if $\hat{s}_k$ is likely to be wrong, then a move from $\hat{\boldsymbol{s}}$ to its neighbor $\boldsymbol{x}$ for the difference position of k should be made. In this case, $\hat{s}_k$ and x_k are neighboring symbols, and $\hat{s}_i = x_i$ for $i \neq k$. In this manner, many of the n_e predicted incorrect symbols can be corrected after $\tau \geq n_e$

searching iterations of the TS algorithm with high probabilities. Let $\bar{\boldsymbol{s}}_{TS}^{\star}$ be the solution found by correcting the predicted incorrect symbols of $\hat{\boldsymbol{s}}$ after τ iterations. We have

$$\bar{\boldsymbol{s}}_{TS}^{\star} = \arg\min_{\bar{\boldsymbol{s}}\in\bar{\mathbb{S}}} \phi(\bar{\boldsymbol{s}}), \tag{4.22}$$

where $\phi(\boldsymbol{x})$ is given in (4.18), and $\bar{\mathbb{S}} = \left\{\bar{\boldsymbol{s}}_{\{1\}}, \ldots, \bar{\boldsymbol{s}}_{\{\tau\}}\right\} \subset \mathbb{S}$, with $\bar{\boldsymbol{s}}_{\{i\}}$ being the current candidate in the ith iteration, $i \leq \tau$.

When n_e is sufficiently small, the complexity involved in finding $\bar{\boldsymbol{s}}_{TS}^{\star}$ with (4.22) becomes much smaller than that for finding $\bar{\boldsymbol{s}}^{\star}$ with (4.21), as well as that for the conventional TS algorithm with (4.16). This is because $\bar{\mathbb{S}}$ is a subset of $\mathbb{S}$, and hence, unlike the conventional TS algorithm, in the ith iteration, $1 \leq i \leq \tau$, a reduced number of neighboring vectors are examined, which implies that the complexity required during τ iterations to find $\bar{\boldsymbol{s}}_{TS}^{\star}$ can be low. If the incorrect symbols are predicted with high accuracy, there is a high probability that $\bar{\boldsymbol{s}}_{TS}^{\star}$ is close to $\hat{\boldsymbol{s}}_{TS}$. Therefore, only a small number of further iterations are required to reach $\hat{\boldsymbol{s}}_{TS}$, and the total complexity of the TS algorithm can be reduced.

3) Proposed ET criteria

In the conventional TS algorithm with ET, the algorithm is terminated early after executing $\mathcal{I}$ iterations, as given in (4.19), where the number of additional iterations is set to $\mathcal{I}_e = \varepsilon\mathcal{I}_{\mathrm{UB}}$ with a fixed value of ε. Remark 4.3 shows that the conventional stopping criterion employing a fixed cutoff factor ε is inefficient. Specifically, in many cases, a further search over $\mathcal{I}_e$ iterations does not yield any performance improvement while creating unnecessary complexities for the TS algorithm. However, if $\mathcal{I}_e$ is set to a small value, the algorithm could be terminated before finding the optimal solution with a high probability, thus causing performance degradation.

In this work, to minimize the number of redundant searching iterations, we propose an adaptive ET criterion based on n_e. We have the following notes:

- If $n_e = 0$, there is no predicted incorrect symbol in $\hat{\boldsymbol{s}}$, which implies a high probability that $\hat{\boldsymbol{s}}$ is already the optimal solution. In this case, no further searching iterations are needed, i.e., $\mathcal{I} = 0$.
- If $n_e \neq 0$ is small, there is a high probability that only a few elements of $\hat{\boldsymbol{s}}$ are incorrect, which can be corrected after a small number of moves in the TS algorithm. Therefore, in this case, only a small $\mathcal{I}$ is required.
- In the case that $n_e \gg 1$, a sufficiently large number of searching iterations should be performed to guarantee the optimal performance.

Therefore, we propose using an adaptive cutoff factor $\hat{\varepsilon}$ depending on $\frac{n_e}{N}$ as follows:

$$\hat{\varepsilon} = \min\left\{\varepsilon, \mu\frac{n_e}{N}\right\}, \tag{4.23}$$

where μ is chosen to guarantee that a sufficient number of iterations is performed in the DL-TS algorithm. Consequently, in the DL-TS algorithm, only $\hat{\mathcal{I}}_e = \hat{\varepsilon}\mathcal{I}_{\mathrm{UB}}$ iterations are required before the searching process is terminated, with $\hat{\mathcal{I}}_e \leq \mathcal{I}_e$ because $\hat{\varepsilon} \leq \varepsilon$. From (4.23), we have $\hat{\mathcal{I}}_e \to 0$ as $n_e \to 0$, and $\hat{\mathcal{I}}_e \to \mathcal{I}_e = \varepsilon\mathcal{I}_{\mathrm{UB}}$ as $n_e \to N$, implying the use of a smaller number of iterations for a smaller n_e, and vice versa.

It is observed from (4.23) that an increase in μ leads to a higher $\hat{\mathcal{I}}_e$. Therefore, more searching iterations are performed in the DL-TS algorithm, which can result in better performance but requires higher complexity. Therefore, in this study, μ is optimized through simulations such that the DL-TS algorithm can achieve almost the same performance as

Algorithm 5 DL-TS detection

Input: $\boldsymbol{H}$, $\boldsymbol{y}$ $\mathcal{I}_{\mathrm{UB}}$, and μ
Output: $\hat{\boldsymbol{s}}_{TS}$

1: Find $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$ based on Algorithm 4.
2: $\boldsymbol{e} = \left|\hat{\boldsymbol{s}} - \hat{\boldsymbol{s}}^{[L]}\right| = [e_1, e_2, \ldots, e_N]^T$
3: $\gamma = \min\{1/\mathrm{SNR}, 0.5\}$.
4: Find $\mathcal{P}$ including the positions of elements of $\boldsymbol{e}$ being larger than γ.
5: Set $n_e = |\mathcal{P}|$.
6: Initialize $\boldsymbol{c} = \hat{\boldsymbol{s}}$ and push $\boldsymbol{c}$ to the tabu list.
7: $\hat{\boldsymbol{s}}_{TS} = \boldsymbol{c}, \phi(\hat{\boldsymbol{s}}_{TS}) = \phi(\boldsymbol{c})$
8: Compute $\hat{\varepsilon}$ based on (4.23).
9: $\hat{\mathcal{I}}_e = \hat{\varepsilon}\mathcal{I}_{\mathrm{UB}}$.
10: $i = 1, count = 0$
11: $\mathcal{S}^{(0)} = \boldsymbol{c}$
12: **while** $i \leq \mathcal{I}_{\mathrm{UB}}$ **and** $count \leq \hat{\mathcal{I}}_e$ **do**
13: **if** $i \leq \tau$ **then**
14: Find $\mathcal{S}(\boldsymbol{c})$ which includes only the neighbors of $\boldsymbol{c}$ with the difference positions in $\mathcal{P}$.
15: $\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\mathcal{S}(\boldsymbol{c})}\{\phi(\boldsymbol{x})\}$
16: **else**
17: Find the neighbor set $\mathcal{N}(\boldsymbol{c})$ of $\boldsymbol{c}$.
18: $\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\mathcal{N}(\boldsymbol{c})}\{\phi(\boldsymbol{x})\}$
19: **end if**
20: Update the current candidate: $\boldsymbol{c} = \boldsymbol{x}^\star$
21: **if** $\phi(\boldsymbol{c}) < \phi(\hat{\boldsymbol{s}}_{TS})$ **then**
22: Update the best solution: $\hat{\boldsymbol{s}}_{TS} = \boldsymbol{c}$.
23: $count = 0$
24: **else**
25: $count = count + 1$
26: **end if**
27: Release the first element in the tabu list if it is full.
28: Push $\boldsymbol{c}$ to the tabu list and update its length.
29: $i = i + 1$
30: **end while**
31: The final solution is the best solution $\hat{\boldsymbol{s}}_{TS}$ found so far.

the conventional TS algorithm with a lower complexity. In other words, μ is chosen as the smallest number that can guarantee marginal performance loss of the DL-TS algorithm.

The DNN-aided TS algorithm is presented in Algorithm 5. In step 1, $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$ are obtained by the FS-Net scheme in Algorithm 4, allowing $\boldsymbol{e}$ to be computed in step 2

as the difference between $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$. In step 4, the list $\mathcal{P}$ including the positions of the predicted incorrect elements of $\hat{\boldsymbol{s}}$ is found based on γ, which is set in step 3. Then, n_e is set to the size of $\mathcal{P}$ in step 5. The DL-TS algorithm is initialized in steps 6–11. Specifically, step 6 assigns $\hat{\boldsymbol{s}}$ to the current candidate $\boldsymbol{c}$ and pushes it to the tabu list. Then, the best solution $\hat{\boldsymbol{s}}_{TS}$ and its metric $\phi\left(\hat{\boldsymbol{s}}_{TS}\right)$ are initialized as $\boldsymbol{c}$ and $\phi(\boldsymbol{c})$, respectively, in step 7. The adaptive cutoff factor $\hat{\varepsilon}$ and $\hat{\mathcal{I}}_e$ are computed in steps 8 and 9, respectively.

In steps 12–30, $\hat{\boldsymbol{s}}_{TS}$ is searched over $\mathcal{I}$ iterations. The first τ iterations are used to correct the predicted incorrect symbols in $\hat{\boldsymbol{s}}$. In particular, in the ith iteration, $1 \leq i \leq \tau$, the best candidate $\boldsymbol{x}^{\star}$ is found in step 15. In contrast, in the remaining $\mathcal{I} - \tau$ iterations, the conventional searching manner is used, where all the neighbors in $\mathcal{N}(\boldsymbol{c})$ are examined to find $\boldsymbol{x}^{\star}$, as in steps 17 and 18. Comparing 15 to 18, it is observed that the proposed searching approach based on the predicted incorrect symbols requires lower complexity than the conventional searching approach because $\mathcal{S}(\boldsymbol{c}) \subset \mathcal{N}(\boldsymbol{c})$. In steps 21–26, if a better solution is found, $\hat{\boldsymbol{s}}_{TS}$ is updated, and at the same time, *count* is set to zero to allow further moves to find a better solution. Otherwise, *count* increases by one until it reaches $\hat{\mathcal{I}}_e$. The following steps update the best solution and the tabu list, and step 31 concludes the final solution after the searching phase is finished.

4.4. Simulation results

In this section, we numerically evaluate the BER performance and computational complexities of the proposed FS-Net detection architecture and the DL-TS algorithm. In our simulations, each channel coefficient is assumed to be an i.i.d. zero-mean complex Gaussian random variable with a variance of $1/2$ per dimension. The SNR is defined as the ratio

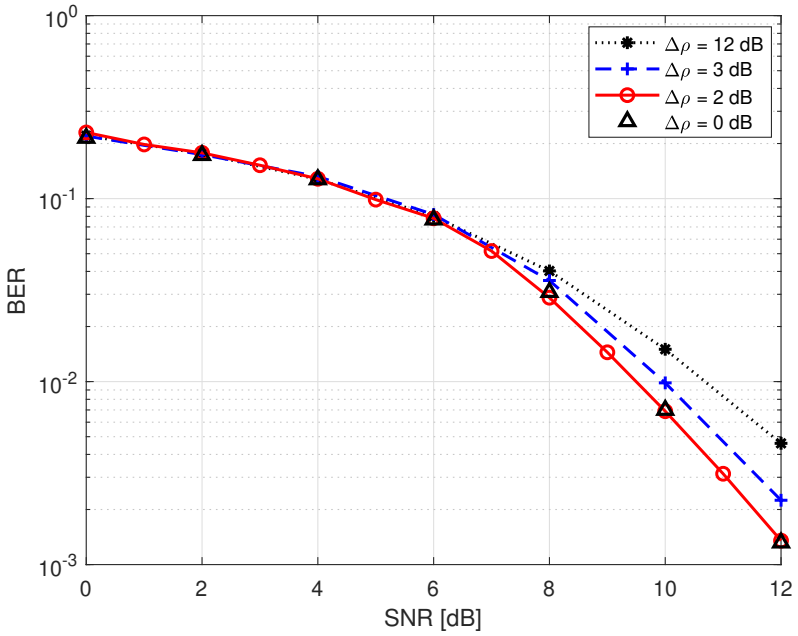


FIG. 4.5. BER performance of the proposed FS-Net architecture for various training SNR ranges corresponding to $\Delta\rho = \{0, 2, 3, 12\}$ dB for $(N_t \times N_r) = (32 \times 32)$, $L = 15$ with QPSK.

$N_t\sigma_t^2/\sigma_n^2$.

4.4.1. Training DNNs

We follow the training model in [31] and [33]. Specifically, we use the Adam optimizer [48], which is a variant of the stochastic gradient descent method [49, 50] for optimizing the DNNs. The DNNs are implemented by using Python with the Tensorflow library [51] and a standard Intel i7-6700 processor. For the training phase, a decaying learning rate of 0.97 and starting learning rate of 0.0001 are used. Furthermore, the number of layers in the DNNs is set depending on the sizes of the considered MIMO systems. We train the DNNs for 35000 iterations, and unless otherwise stated, batch sizes of 2000 samples are used. For each sample, $\boldsymbol{s}, \boldsymbol{H}$, and $\boldsymbol{y}$ are independently generated from (1.2), then $\boldsymbol{H}^T\boldsymbol{H}$ and $\boldsymbol{H}^T\boldsymbol{y}$ are computed accordingly.

In the training phase of the DetNet and ScNet, the noise variance can be randomly generated so that the SNR is uniformly distributed in the range $[\rho_{min}, \rho_{max}]$. In this training method, the network needs to adapt to a wide range of SNR. Therefore, a large number of layers and neurons need to be employed, which requires high computational complexity and a long training time. To resolve this problem, we consider training multiple networks for multiple divided SNR ranges. The only disadvantage of this method is its requirement of additional memory to save multiple trained models. However, the network can learn faster to achieve higher accuracy.

To show the performance improvement by reducing the SNR range in Fig. 4.5, we train and test the FS-Net for 32×32 MIMO with QPSK and $\Delta\rho = \{0, 2, 3, 12\}$ dB with $\beta = 0.5$ being set for the loss function in (4.11). Here, $\Delta\rho = \rho_{max} - \rho_{min}$ represents the SNR interval for which a DNN is trained. For example, $\Delta\rho = 0$ dB means that each FS-Net is trained only for a particular SNR. In contrast, for $\Delta\rho = 12$ dB, it is trained for the entire considered SNR range as the DetNet and ScNet are trained in [31–33]. In each training epoch, the SNR is randomly selected in the corresponding range. For performance evaluation, an appropriately trained FS-Net is selected for each considered SNR. For example, to test the FS-Net at SNR = 10 and 11 dB, the FS-Net trained for the SNR range of $[9, 11]$ dB is employed.

The BER performance for $\Delta\rho = \{0, 2, 3, 12\}$ dB is shown in Fig. 4.5. It is observed that as $\Delta\rho$ decreases, the BER performance is significantly improved. In particular, the FS-Nets with $\Delta\rho = 0$ and 2 dB achieve the best BER performance. For realistic environments with arbitrary SNRs, the DNN trained only for a specific SNR, i.e., $\Delta\rho = 0$ dB, can be impractical; therefore, for the remaining simulations, we train the DetNet, ScNet, and FS-Net with $\Delta\rho = 2$ dB. We note that for different SNR ranges, the trained DNNs have

different weights and biases but the same network architecture. Therefore, they do not result in any further computational complexity in the operations of the FS-Net and DL-TS algorithm.

4.4.2. Performance and complexity of the proposed FS-Net

In Fig. 4.6, we show the BER performance of the proposed FS-Net in comparison to those of the DetNet [31] and ScNet [33] for two scenarios: $N_t < N_r$ and $N_t = N_r$. Specifically, we assume $(N_t \times N_r) = (32 \times 64)$ in Fig. 4.6A and $(N_t \times N_r) = (32 \times 32)$ in Fig. 4.6B, and $L = 15$ and QPSK are assumed in both scenarios. Furthermore, the DNNs in Fig. 4.6A are trained with batch sizes of 3000 samples. For comparison, linear ZF/MMSE, ZF-based OSIC schemes [52], Twin-DNN model [53], and SE-SD [47] are also considered. From Fig. 4.6, the following observations are noted:

- For both considered scenarios, the DetNet, ScNet, and FS-Net schemes outperform the linear ZF and MMSE receivers. The OSIC receiver only performs better than the DetNet and ScNet schemes for SNR $>$ 8 dB in the 32×64 MIMO system. However, in the 32×32 MIMO system, the OSIC receiver is outperformed by the DNN-based schemes. The performance of the Twin-DNN model is close to that of the ScNet, and it achieves performance improvement with respect to the conventional DetNet only in a 32×32 MIMO system.

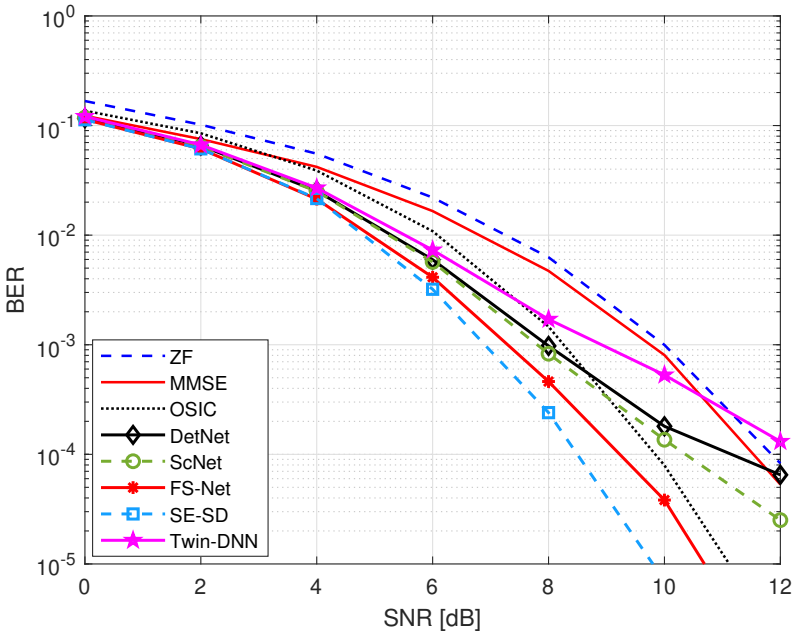
- It is clear that among the compared low-complexity schemes, including the linear ZF/MMSE, OSIC receivers, and DNN detection architectures, the proposed FS-Net achieves the best performance in both considered scenarios. Specifically, as seen in Fig. 4.6A, it achieves SNR gains of approximately 2 dB and 3 dB with respect to the ScNet and DetNet architectures in the high-SNR region, respectively, and the

TABLE 4.2. Computational complexities of the DetNet, ScNet, and FS-Net for 32×64 and 32×32 MIMO with QPSK and $L = 15$.

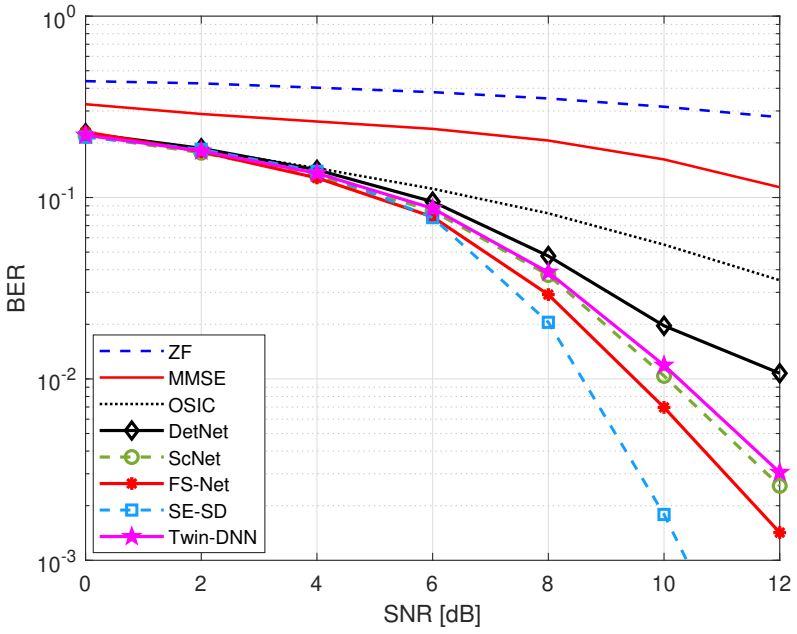
Complexity	DetNet	ScNet	FS-Net
32×64 MIMO	2.168×10^7	1.188×10^6	1.187×10^6
32×32 MIMO	1.627×10^6	6.476×10^5	6.467×10^5

gains in Fig. 4.6B are approximately 1 dB and 2 dB over those of ScNet and DetNet, respectively. Compared to the Twin-DNN model, the proposed FS-Net achieves SNR gains of approximately 3 and 1 dB in Figs. 4.6A and 4.6B, respectively.

In Table 4.2 and Fig. 4.7, we show the complexities of the DetNet, ScNet, and FS-Net schemes, which are computed based on (4.12)–(4.15). The same simulation parameters as in Fig. 4.6 are used for Table 4.2. By contrast, in Fig. 4.7, we consider $N_t = N_r \in [16, 64]$, $L = 15$, with QPSK and 16-QAM to show the complexities of the examined DNN architectures as the system size increases. It is seen from Figs. 4.6, 4.7 and Table 4.2 that the proposed FS-Net architecture not only achieves better BER performance but also requires lower computational complexity than the DetNet and ScNet schemes. Specifically, the complexity of the FS-Net is much lower than that of the DetNet and slightly lower than that of the ScNet. Therefore, we can conclude that among the compared architectures, the proposed FS-Net achieves the best performance–complexity tradeoff. Although the SE-SD receiver outperforms the FS-Net, as seen in Fig. 4.6, it requires high computational complexity, which is even higher than that of the conventional TS algorithm in large MIMO systems [35]. Therefore, among the compared schemes, the FS-Net is chosen as a scheme for generating the initial solution of the DL-TS algorithm, as presented in Section 4.3.2.



(A) $(N_t \times N_r) = (32 \times 64)$, $L = 15$



(B) $(N_t \times N_r) = (32 \times 32)$, $L = 15$

FIG. 4.6. BER performance of the proposed FS-Net architecture in comparison with those of the DetNet, ScNet, Twin-DNN, ZF, MMSE, OSIC, and SE-SD schemes with QPSK.

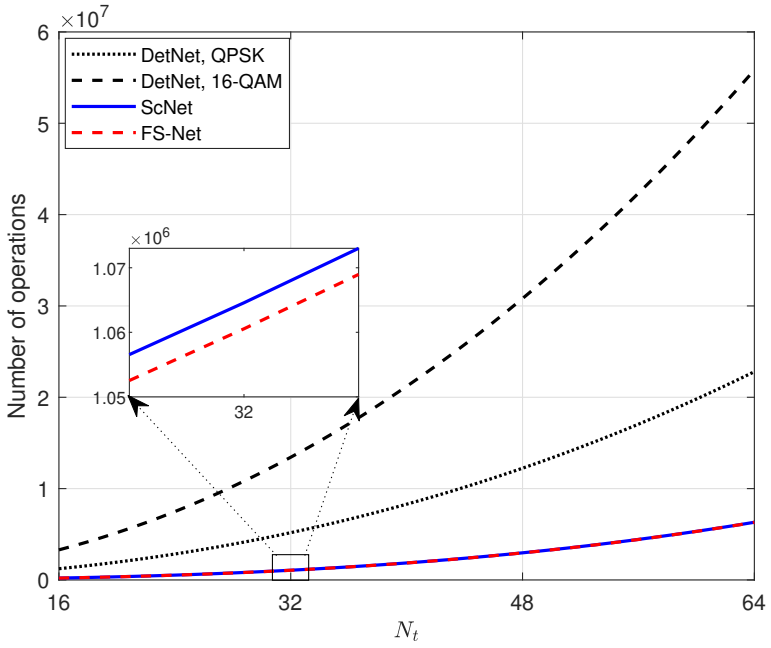


FIG. 4.7. Complexities of the DetNet, ScNet, and FS-Net with $N_t = N_r \in [16, 64]$, $L = 15$, QPSK, and 16-QAM.

TABLE 4.3. Simulation parameters used in Figs. 4.8 and 4.9

Parameters	32×32, QPSK	8×8, 16-QAM
$\mathcal{I}_{\text{UB}}$	800	2500
ε	0.4	0.4
μ	4	5
L	15	20

4.4.3. Performance of the proposed DL-TS algorithm

In this section, we show the BER performance of the proposed DL-TS algorithm, which is compared to those of the SE-SD scheme [47] and the conventional TS algorithm where ET is not applied [27], and the ZF solution is used as the initial solution. Furthermore, to show the advantage of the proposed DL-TS algorithm, it is also compared to the TS schemes with ET and ZF (ZF-TS), MMSE (MMSE-TS), OSIC (OSIC-TS), and ScNet (ScNet-TS) for initial solutions.

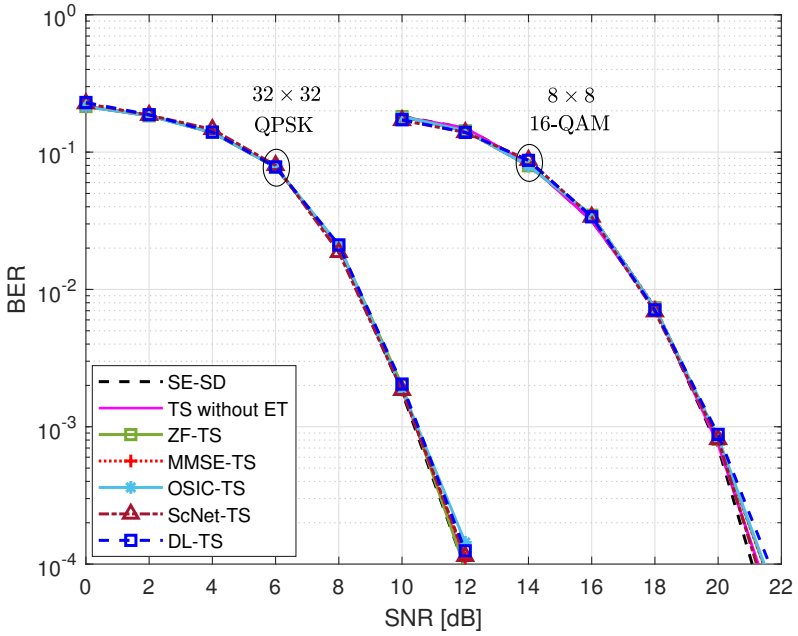


FIG. 4.8. BER performance of the proposed DL-TS algorithm in comparison with those of the ZF-TS, MMSE-TS, OSIC-TS, ScNet-TS, TS without ET, and the SE-SD receivers for $(N_t \times N_r) = (32 \times 32)$, $L = 15$ with QPSK and $(N_t \times N_r) = (8 \times 8)$, $L = 20$ with 16-QAM.

For comparison, various systems are considered, including the 32×32 MIMO with QPSK and 8×8 MIMO with 16-QAM, and the simulation parameters are presented in Table 4.3. Specifically, $\mathcal{I}_{\text{UB}}$ is set based on [35], which guarantees that the TS schemes can approximately achieve the BER performance of the SE-SD receiver. The length of the tabu list is set to $P = \mathcal{I}_{\text{UB}}/2$ so that TS algorithms approximately approach the optimal performance [27, 35]. In all the considered systems, we use $\varepsilon = 0.4$, for which the employment of ET results in only marginal performance loss. Furthermore, for the DL-TS algorithm, μ ($\leq \varepsilon N$) is optimized through simulations, and $\tau = 4n_e$ is assumed for the number of searching iterations in which only the predicted incorrect symbols are considered for neighbor search. Regarding the FS-Net architecture, the number of layers, L, should be chosen such that the FS-Net-based solution can be found with a good performance–complexity/training time trade-off. For this, the FS-Net for each MIMO systems is trained

and tested with different values of L, and the chosen ones are presented in Table 4.3.

It is observed from Fig. 4.8 that in all the considered systems, the TS algorithms with ET, namely, ZF-TS, MMSE-TS, OSIC-TS, ScNet-TS, and DL-TS, all have approximately the same BER performance. In particular, although ET is employed, only marginal performance losses are seen for the ZF-TS, MMSE-TS, OSIC-TS, ScNet-TS, and DL-TS schemes with respect to the conventional TS algorithm without ET. Furthermore, with the chosen parameters, all the considered TS algorithms approximately achieve the performance of the SE-SD scheme.

4.4.4. Complexity reduction of the DL-TS algorithm

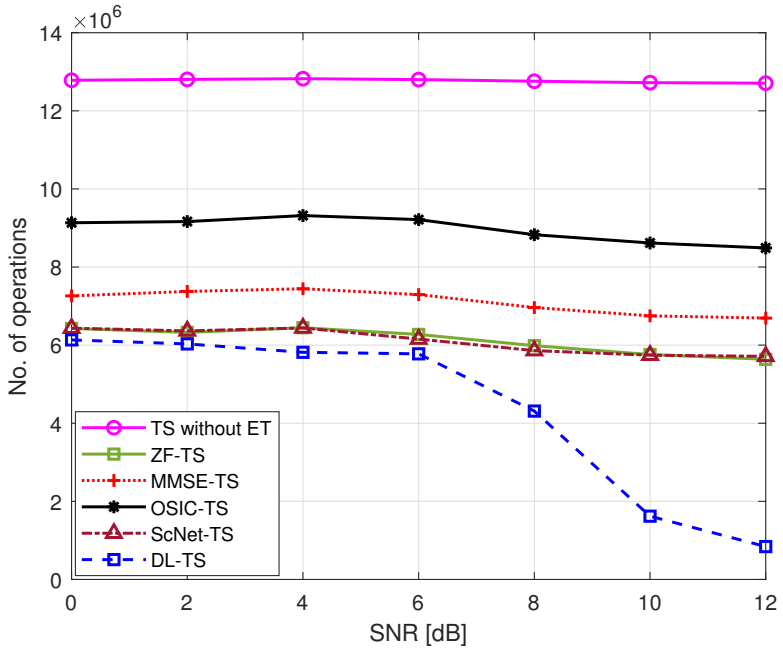
In Fig. 4.9, the complexity of the DL-TS algorithm is compared to those of the conventional TS, ZF-TS, MMSE-TS, OSIC-TS, and ScNet-TS for 32×32 MIMO system with QPSK and 8×8 MIMO with 16-QAM. To ensure that the compared algorithms have approximately the same BER performance, the simulation parameters are assumed to be exactly the same as those in Fig. 4.8 and are presented in Table 4.3. It is clear from Fig. 4.9 that the TS algorithms with ET can significantly reduce the complexity of the TS algorithm, while resulting in only marginal performance losses, as discussed in the previous subsection. Furthermore, among the TS algorithms with ET, the DL-TS scheme has the lowest computational complexity. For example, in Fig. 4.9A, at SNR = 12 dB, the complexity of the DL-TS algorithm is only approximately 14.9%, 14.7%, 12.6%, 9.9%, and 6.6% those of the ZF-TS, ScNet-TS, MMSE-TS, OSIC-TS, and conventional TS scheme, respectively. In Fig. 4.9B, the complexity-reduction ratio of the proposed DL-TS algorithm is approximately 54.2% with respect to the other ET-based TS algorithms and 84.8% with respect to the conventional TS without ET at SNR = 22 dB. In particular, Fig. 4.9 shows

that the complexity reduction of the ScNet-TS scheme is not as significant as that of the DL-TS algorithm. The reason is that although the ScNet can generate better initial solutions for the TS algorithm, it has higher complexity than the linear and OSIC receivers. Therefore, its complexity reduction in the iterative searching phase cannot compensate for the complexity increase in the initialization phase.

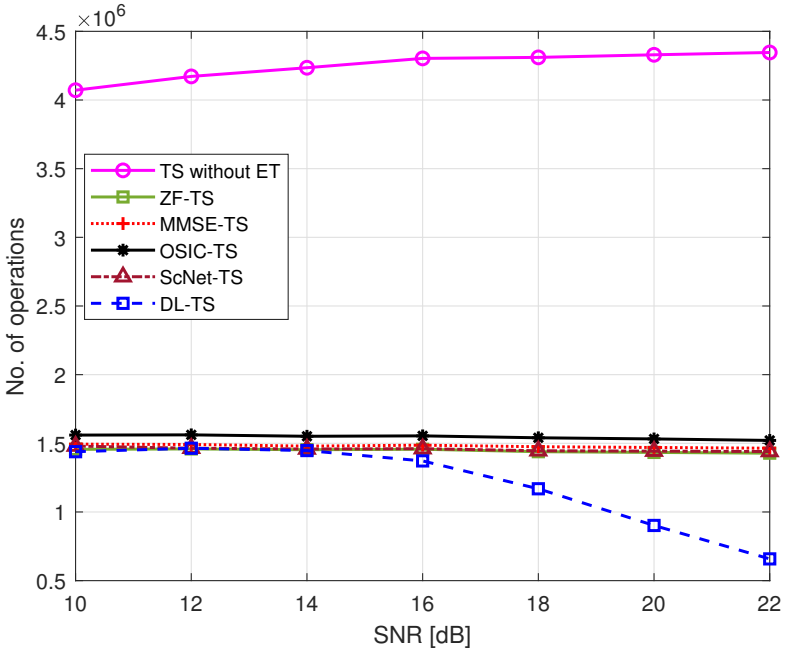
Furthermore, we see that unlike the ZF-TS, MMSE-TS, OSIC-TS, and conventional TS schemes, the complexity of the proposed DL-TS algorithm decreases with SNR. The reason for that is as follows:

- At low SNRs, the FS-Net scheme in Algorithm 4 has low accuracy, which leads to large n_e. In this case, we have $\hat{\varepsilon} \approx \varepsilon$, and hence, the complexity reduction is only obtained by reduced search space during the first iterations. However, for large n_e, the reduction of search space becomes small, which leads to a relatively small gain in complexity reduction.

- At high SNRs, the FS-Net-based initial solution is generated with high accuracy and small n_e. Therefore, a significant complexity reduction is achieved with the DL-TS algorithm owing to a reduced cutoff factor $\hat{\varepsilon}$ as well as the reduced search space during the first τ iterations.



(A) $(N_t \times N_r) = (32 \times 32)$, QPSK, $L = 15$



(B) $(N_t \times N_r) = (8 \times 8)$, 16-QAM, $L = 20$

FIG. 4.9. Complexity of the proposed DL-TS algorithm in comparison with those of the ZF-TS, MMSE-TS, OSIC-TS, ScNet-TS, and TS without ET.

4.5. Conclusion

In this study, we have presented the FS-Net detection scheme, which is a DNN-aided symbol-detection algorithm for MIMO systems. Unlike the prior DetNet and ScNet schemes, the input vector of the FS-Net is optimized such that a reduced number of connections is required in each layer. Furthermore, the correlation between the input and output signals is considered in the loss function of the FS-Net, allowing it to be trained better with faster convergence compared to DetNet and ScNet. The optimized input, network connection, and loss function lead to the reduced complexity and improved performance of the proposed FS-Net scheme.

The proposed FS-Net is then incorporated into the TS algorithm. Specifically, it is used to generate the initial solution with enhanced accuracy for the TS algorithm. Furthermore, based on the initial solution given by the FS-Net, the iterative searching phase of the TS algorithm is improved by predicting incorrect symbols. This can facilitate more efficient moves in the TS algorithm so that the optimal solution is more likely to be reached earlier. We also propose using an adaptive ET criterion, in which the cutoff factor is adjusted based on the accuracy of the FS-Net-based initial solution. As a result, a small number of searching iterations is taken for a reliable initial solution, which leads to a reduction in the overall complexity of the TS algorithm.

Our simulation results show that the proposed FS-Net scheme not only outperforms the linear ZF/MMSE and the OSIC receivers but also achieves improved performance and reduced complexity with respect to the DetNet and ScNet. Furthermore, with almost the same BER performance, the proposed DL-TS scheme with the FS-Net-based initial solution requires much lower complexity than the other TS algorithms with ET, such as

ZF-TS, MMSE-TS, OSIC-TS, and ScNet-TS. We note that the proposed DL-TS scheme can be combined with existing TS algorithms, such as LTS [10], R3TS [26], and QR-TS [35], for better initialization and more efficient ET. In our simulations, the Rayleigh fading channel is used for training and testing the performance of the considered DNNs. For future work, to consider various practical communication scenarios, an online training method can be considered that retrains the DNNs such that they adapt to the new channel environment.

5. Application of Deep Learning to Sphere Decoding for Large MIMO Systems

5.1. Motivation and contribution

The sequential SD with reduced complexity and near-optimal performance w.r.t the optimal ML detector has been optimized well for small- and moderate-size MIMO systems. Among its variants, the SE-SD [47] has the same performance as the conventional FP-SD [54] with reduced complexity. However, its complexity remains very high in large MIMO systems [35]. To address the problems of sequential SD, KSD [5] was proposed to achieve fixed and reduced complexity. However, this algorithm suffers performance degradation, and does not ensure complexity reduction at high SNRs.

The aforementioned challenges of the sequential SD and KSD make them infeasible for large MIMO systems. However, the increasing application of DL in wireless communication creates room for further optimization of SD schemes. Particularly, the initial works on DL-aided SD in [38] and [34] attempt to improve SD by employing a DNN to learn the initial radius. Whereas a single radius (SR) is used in [38], multiple radii (MR) are employed in [34]. In this study, to distinguish them, we refer to the former as the SR-DL-SD scheme and to the latter as the MR-DL-SD scheme. Furthermore, as an improvement of [38] and [34], Weon *et al.* [39] propose a learning-aided deep path-prediction scheme for sphere

decoding (DPP-SD) in large MIMO systems. Specifically, the minimum radius for each sub-tree is learned by a DNN, resulting in more significant complexity reduction w.r.t. the prior SR-DL-SD and MR-DL-SD schemes.

In all three DL-aided SD schemes mentioned above, the common idea is to predict radii for the sequential SD. This approach has some limitations in the offline learning phase, as well as during online application to SD. First, in the DNN training phase in [34, 38, 39], conventional SD needs to be performed first to generate training labels, i.e., the radius. Consequently, time and computational complexity requirements are high to train these DNNs. Although the training phase can be performed offline, these time and resource requirements make such schemes less efficient. Second, although the radius plays an important role in the search efficiency of conventional FP-SD, it becomes less significant in SE-SD [54]. Specifically, in the SE-SD scheme, the radius of the sphere can be initialized to a sufficiently large value; the sphere then shrinks whenever a valid lattice point is found. Therefore, using the predicted radius becomes less efficient in SE-SD, especially for high SNRs, for which a relatively reliable radius can be computed using the conventional formula [54]. Moreover, in the KSD, the breadth-first search does not require a radius, which implies that the learning objectives in [34,38,39] are inapplicable to the KSD scheme.

In this chapter, we propose the FDL-SD and FDL-KSD algorithms, which can overcome the limitations of the existing DL-aided SD schemes via a novel application of DL to SD. Specifically, we use a DNN to generate a highly reliable initial candidate for the search in SD, rather than generating the radius as in the existing DL-aided SD schemes [34, 38, 39]. Furthermore, the output of the DNN facilitates a candidate/layer-ordering scheme and an early rejection scheme to significantly reduce the complexity. The proposed ideas are applicable to both sequential SD and KSD. Our specific contributions can be summarized

as follows:

- We propose applying FS-Net, a DNN architecture that was introduced in [37] to achieve an improved tradeoff between the BER performance and computational complexity, to generate an initial candidate for SD. Unlike other architecture that uses DNNs for learning the radius [34, 38, 39], the FS-Net can be trained easily without performing conventional SD; this considerably reduces the time and computational resources required for the training phase.

- We propose the FDL-SD scheme, which achieves significant complexity reduction while fully preserving the performance of the conventional SD. Specifically, we exploit the output of the FS-Net to facilitate the search in SD based on the following ideas:

  (i) First, the output of the FS-Net, which is the approximate of the transmitted signal vector, is employed to determine the search order in the SD scheme. In particular, the candidates are ordered such that those closer to the FS-Net's output are tested first. This approach enhances the chance that the optimal solution is found early and accelerates the shrinking of the sphere, resulting in complexity reduction of the proposed FDL-SD scheme.

  (ii) Second, we propose a layer-ordering scheme. Specifically, we found that the sequential tree-like search in SD can be considered as the process of exploring and correcting incorrectly detected symbols. This implies that the errors at the lower layers can be explored and corrected sooner. Motivated by this, we propose ordering the layers of candidates so that errors are more likely to occur at low layers. This order is determined based on the FS-Net's output.

- In the proposed FDL-KSD scheme, the FS-Net's output is also leveraged to optimize the search process of the KSD. In this scheme, we employ the cost metric of the

FS-Net-based solution as a threshold to reject unpromising candidates early. Furthermore, the layer ordering in *(ii)* is used to reduce the chance that the optimal solution is rejected early. This results in not only performance improvement, but also complexity reduction w.r.t. the conventional KSD.

- Our extensive simulation results show that the FDL-SD scheme achieves a remarkable complexity reduction without any performance loss w.r.t. the conventional SD. In particular, the complexity reduction attained by our proposed FDL-SD scheme is significantly greater than those acquired by the existing DL-aided SD schemes. Furthermore, the proposed FDL-KSD scheme exhibits a considerable improvement in the performance–complexity tradeoff w.r.t. the conventional KSD.

The rest of this chapter is organized as follows: In Section 5.2, the existing DNNs for MIMO detection are reviewed. The proposed FDL-SD and FDL-KSD are presented in Sections 5.3 and 5.4, respectively. In Section 5.5, the simulation results and numerical discussions are presented. Finally, the conclusions are drawn in Section 5.6.

5.2. Why FS-Net?

A number of DNNs have been designed for symbol detection in large MIMO systems [31–33,37,55,56]. Specifically, Samuel *et al.* in [31] and [32] introduced the first DNN-based detector, called DetNet. However, the DetNet performs poorly for large MIMO systems with $N \approx M$; it also has a complicated network architecture with high computational complexity. To overcome these challenges, ScNet [33] and FS-Net [37] are proposed. They simplify the network architecture and improve the loss function of the DetNet, which lead to significant performance improvement and complexity reduction. Shortly afterwards, by

unfolding the orthogonal approximate message passing (OAMP) algorithm [57], He *et al.* introduced the OAMP-Net [55] for symbol detection in both i.i.d. Gaussian and small-size correlated channels. Furthermore, Khani *et al.* in [56] focus on realistic channels and propose the MMNet, which significantly outperforms the OAMP-Net with the same or lower computational complexity. We note that all the DNNs mentioned here are unfolding architectures, unlike the well-known FC-DNN. The ability of the FC-DNN for symbol detection is investigated in [31]; the investigation shows that, despite performing well for fixed channels, FC-DNN cannot detect symbols when the channel varies.

The main application of DL to SD in this work is to generate a highly reliable candidate $\hat{\boldsymbol{s}}$ that is an approximate of the transmitted signal vector $\boldsymbol{s}$. This can be achieved by any of the aforementioned DNNs, i.e., DetNet, ScNet, FS-Net, OAMP-Net, and MMNet. In this work, we choose FS-Net because of its low complexity and good BER performance. Specifically, among the discussed DNNs, the OAMP-Net has the highest computational complexity because pseudo-matrix inversion is performed in each layer to conduct the linear MMSE estimation [55]. In the MMNet, high complexity is required to compute the standard deviation of the Gaussian noise on the denoiser inputs in each layer [56]. Meanwhile, the DetNet deploys the dense-connection architecture with high-dimensional input vectors in every layer [31,32], which causes an extremely high computational load. In contrast, with its simplified network connections and optimized loss function, the FS-Net achieves significant performance improvement while requiring much lower computational complexity than the DetNet and ScNet [37]. Therefore, it is chosen for incorporation with SD schemes in this work.

5.3. Proposed FDL-SD scheme

We first briefly review the common ideas of SD based on the description in [54]. Some notations in [54] are also adopted for ease of notation. Similar to the ML detection, SD attempts to find the optimal lattice point that is closest to $\boldsymbol{y}$, but its search is limited to the points inside a sphere of radius d, i.e.,

$$\hat{\boldsymbol{s}}_{SD} = \arg \min_{\boldsymbol{x} \in \mathcal{S} \subset \mathcal{A}^N} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2 , \tag{5.1}$$

where $\mathcal{S}$ is a hypersphere specified by the center $\boldsymbol{y}$ and radius d. Each time a valid lattice point is found, the search is further restricted, or equivalently, the sphere shrinks, by decreasing the radius, as illustrated in Fig. 5.1A. In this way, when there is only one point in the sphere, the point becomes the final solution $\hat{\boldsymbol{s}}_{SD}$. The ingenuity of SD is the identification of the lattice points that lie inside the sphere, which is discussed below.

A lattice point $\boldsymbol{H}\boldsymbol{x}$ lies inside a sphere of radius d if and only if $\boldsymbol{x}$ fulfills condition $(\mathcal{C}) : \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2 \leq d^2$. In SD, the QR decomposition of the channel matrix is useful in breaking $(\mathcal{C})$ into the necessary conditions for each element of $\boldsymbol{x}$. Let

$$\boldsymbol{H} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R} \\ \mathbf{0}_{(M-N)\times N} \end{bmatrix},$$

where $\boldsymbol{Q} = [\boldsymbol{Q}_1 \;\; \boldsymbol{Q}_2]$ is an $M \times M$ unitary matrix, having the first N and last $M - N$ orthonormal columns in $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$, respectively. $\boldsymbol{R}$ is an $N \times N$ upper triangular matrix, and $\mathbf{0}_{(M-N)\times N}$ represents a matrix of size $(M - N) \times N$ containing all zeros. Applying QR decomposition, $(\mathcal{C})$ can be rewritten as $\|\boldsymbol{z} - \boldsymbol{R}\boldsymbol{x}\|^2 \leq d_N^2$, where $\boldsymbol{z} = \boldsymbol{Q}_1^T \boldsymbol{y}$ and

$d_N^2 = d^2 - \left\| \boldsymbol{Q}_2^T \boldsymbol{y} \right\|^2$. Owing to the upper-triangular structure of $\boldsymbol{R}$, we have

$$(\mathcal{C}) : \sum_{n=1}^{N} \left(z_n - \sum_{i=n}^{N} r_{n,i} x_i \right)^2 \leq d_N^2.$$

Consequently, the necessary conditions for the elements of $\boldsymbol{x}$ to fulfill $(\mathcal{C})$ can be expressed as

$$LB_n \leq x_n \leq UB_n, n = 1, 2, \ldots, N, \tag{5.2}$$

where x_n represents the nth element of $\boldsymbol{x}$. In (5.2), LB_n and UB_n respectively denote the lower and upper bounds of x_n. Without loss of generality, we assume that the entries on the main diagonal of $\boldsymbol{R}$ are positive, i.e., $r_{n,n} > 0, n = 1, \ldots, N$. Then, LB_n and UB_n can be given as

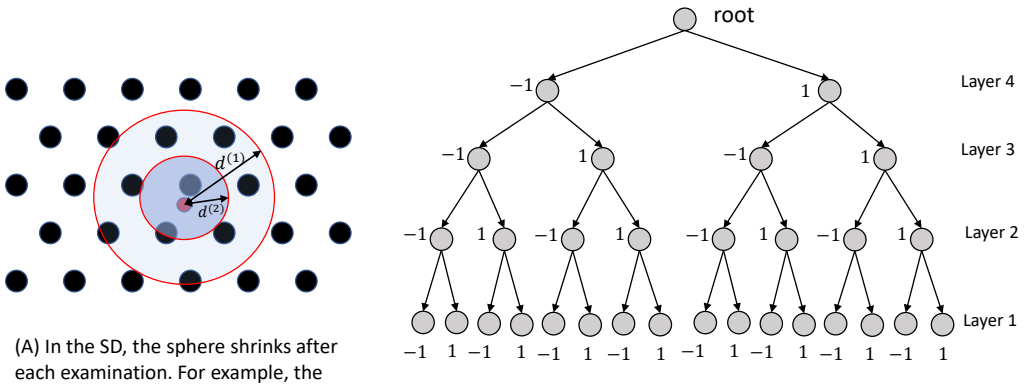
$$LB_n = \left\lceil \frac{z_{n|n+1} - d_n}{r_{n,n}} \right\rceil, \; UB_n = \left\lfloor \frac{z_{n|n+1} + d_n}{r_{n,n}} \right\rfloor, \tag{5.3}$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ round a value to its nearest larger and smaller symbols in alphabet $\mathcal{A}$, respectively, and

$$z_{n|n+1} = \begin{cases} z_N, & n = N \\ z_n - \sum_{i=n+1}^{N} r_{n,i} x_i, & n < N \end{cases} \tag{5.4}$$

means adjusting z_n based on the chosen symbols of $\boldsymbol{x}$, i.e., $\{x_{n+1}, x_{n+2}, \ldots, x_N\}$. Furthermore, d_n in (5.3) is given by

$$d_n^2 = \begin{cases} d_N^2, & n = N \\ d_{n+1}^2 - \sigma_{n+1}^2, & n < N \end{cases}, \tag{5.5}$$



(A) In the SD, the sphere shrinks after each examination. For example, the radius reduces from $d^{(1)}$ in the first search to $d^{(2)}(< d^{(1)})$ in the second search.

(B) Tree-like model representation of the SD algorithm with $N = 4$ and QPSK.

FIG. 5.1. Illustration of the SD scheme with the lattice and tree-like model.

where

$$\sigma_{n+1}^2 = \left(z_{n+1|n+2} - r_{n+1,n+1}x_{n+1}\right)^2. \tag{5.6}$$

The tree-like model is useful to illustrate the candidate exploration and examination in SD schemes. It maps all possible candidates $\boldsymbol{x} \in \mathcal{A}^N$ to a tree with N layers, each associated with an element of $\boldsymbol{x}$. Layer n of the tree has nodes representing possibilities for x_n. A candidate is examined by extending a path, starting from the root, over nodes in the layers. When the lowest layer is reached, a complete path represents a candidate $\boldsymbol{x}$. As an example, a tree-like model for a MIMO system with $N = 4$ and QPSK is illustrated in Fig. 5.1B, which has four layers, corresponding to $N = 4$ elements of a candidate, and $|\mathcal{A}|^N = 16$ complete paths, representing 16 candidates for the solution, where $|\mathcal{A}| = 2$ for QPSK signals. Based on the tree-like model, in the sequential SD, the candidates are explored in the depth-first search strategy. Specifically, the algorithm explores the nodes associated with the symbols satisfying (5.2) from the highest to lowest layers. Once the lowest layer of a candidate $\boldsymbol{x}$ is reached, a valid lattice point is found, and the radius is

reduced to $\phi(\boldsymbol{x}) < d$, where $\phi(\boldsymbol{x}) = \|\boldsymbol{z} - \boldsymbol{R}\boldsymbol{x}\|^2 = d_N^2 - d_1^2 + (z_1 - r_{1,1}x_1)^2$ is the ML metric of $\boldsymbol{x}$ [54]. Based on this search procedure, we found that SD can be optimized by ordering the examined candidates and layers in conjunction with the output of the FS-Net, as presented in the following subsections.

5.3.1. Candidate ordering

The complexity of the SD scheme significantly depends on the number of lattice points that lie inside the sphere, or equivalently, the number of candidates that need to be examined. The sphere shrinks after a valid lattice point is found. Therefore, it is best to start the search by examining an optimal or near-optimal point. In the best case, if the algorithm starts with the optimal solution, i.e., $\boldsymbol{x} = \hat{\boldsymbol{s}}_{ML}$, the radius can decrease rapidly to $\phi(\hat{\boldsymbol{s}}_{ML})$. As a result, no more lattice points lie inside the newly shrunken sphere, and the solution is concluded to be $\hat{\boldsymbol{s}}_{SD} = \hat{\boldsymbol{s}}_{ML}$. However, finding the optimal point requires high computational complexity, which is as challenging as performing the SD scheme itself. Furthermore, the simple linear ZF/MMSE or SIC detector cannot guarantee a highly reliable solution in practical multiuser large MIMO systems. Therefore, we propose using a DNN to find a reliable candidate for initializing the search in SD. For the reasons explained in Section 5.2, the FS-Net is employed for this purpose.

In the proposed FDL-SD scheme, the search starts by examination of $\hat{\boldsymbol{s}}$ obtained in step 7 of Algorithm 4. Furthermore, as $\hat{\boldsymbol{s}}^{[L]}$ is the output of the FS-Net, it is natural to perform the search in an order such that a candidate closer to $\hat{\boldsymbol{s}}^{[L]}$ is examined earlier. To this end, the symbols satisfying (5.2) in layer n are ordered by increasing distance from $\hat{s}_n^{[L]}$,

$n = 1, 2, \ldots, N$. Specifically, in layer n, the symbols are examined in the order

$$\mathcal{O}_n^{\mathrm{FDL}} = \{\hat{s}_{n,1}, \hat{s}_{n,2}, \ldots\}, \text{where } \hat{s}_{n,i} \in \mathcal{A} \text{ and } LB_n \leq \hat{s}_{n,i} \leq UB_n, \tag{5.7}$$

with $\hat{s}_{n,i}$ being the ith-closest symbol to $\hat{s}_n^{[L]}$ and LB_n and UB_n being given in (5.3). By using $\{\mathcal{O}_1^{\mathrm{FDL}}, \mathcal{O}_2^{\mathrm{FDL}}, \ldots, \mathcal{O}_N^{\mathrm{FDL}}\}$, the first candidate examined in the FDL-SD scheme is $\hat{\boldsymbol{s}}$. It can be seen that in this scheme, the initial sphere is predetermined by the radius $\phi(\hat{\boldsymbol{s}})$. Therefore, optimization of the initial radius is not required in the proposed FDL-SD scheme.

It is worth noting that the proposed candidate ordering in the FDL-SD scheme is different from that in the SE-SD scheme. Specifically, the SE-SD scheme examines the candidate nearest to the center of the sphere first, then moves outward to the surface of the sphere [58]. Therefore, the order $\mathcal{O}_n^{\mathrm{SE}}$ is employed in the SE-SD scheme [58, 59], which is given by

$$\mathcal{O}_n^{\mathrm{SE}} = \{\bar{s}_{n,1}, \bar{s}_{n,2}, \ldots\}, \text{where } \bar{s}_{n,i} \in \mathcal{A} \text{ and } LB_n \leq \bar{s}_{n,i} \leq UB_n. \tag{5.8}$$

Here, $\bar{s}_{n,i}$ is the ith closest symbol to $\bar{s}_n = \frac{z_{n|n+1}}{r_{n,n}}$ and LB_n and UB_n are given in (5.3). Our simulation results show that the proposed FDL-SD scheme with order $\mathcal{O}_n^{\mathrm{FDL}}$ results in considerable complexity reduction, which is much more significant than that provided by the SE-SD with order $\mathcal{O}_n^{\mathrm{SE}}$.

5.3.2. Layer ordering

In the SD scheme, once a candidate $\boldsymbol{x}$ is examined, the next step is to search for a better solution in the shrunken sphere. This is equivalent to the process of correcting incorrect

symbols in $\boldsymbol{x}$, and the faster a wrong symbol is corrected, the earlier the optimal solution is found, which results in lower complexity of the SD scheme. Furthermore, knowledge of the positions of erroneous symbols can significantly affect the efficiency of error correction. Therefore, in this subsection, we are interested in optimizing the layers of erroneous symbols, which will be referred to as *erroneous layers* from now on.

Similar to the conventional SD, in the proposed FDL-SD, the search repeatedly moves downward and upward over layers to explore candidates. Given that $\hat{\boldsymbol{s}}$ is examined first and the order $\mathcal{O}_n^{\mathrm{FDL}}$ in (5.7) is used in each layer, the candidates are examined in the following order:

$$\begin{bmatrix} \hat{s}_N \\ \vdots \\ \hat{s}_{1,1} \end{bmatrix} \rightarrow \begin{bmatrix} \hat{s}_N \\ \vdots \\ \hat{s}_{1,2} \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} \hat{s}_{N,1} \\ \vdots \\ \hat{s}_1 \end{bmatrix} \rightarrow \begin{bmatrix} \hat{s}_{N,2} \\ \vdots \\ \hat{s}_1 \end{bmatrix} \rightarrow \ldots,$$

where the layers of candidates are reversed to reflect the tree-model-based search strategy in SD. From the candidate examination order above, we have a note in the following remark.

Remark 5.1. If an erroneous layer is $\tilde{n}$, i.e., $\hat{s}_{\tilde{n}} \neq s_{\tilde{n}}$, $\hat{s}_{\tilde{n}}$ can be corrected by replacing it with one of the other symbols in $\mathcal{O}_n^{\mathrm{FDL}}$. In particular, the lower the erroneous layer is, the faster the corresponding erroneous symbol is corrected. An equivalent interpretation using the tree-like model is that, if there is an erroneous node, the closer to the leaf nodes it is, the faster it is corrected. For example, if $\tilde{n} = 1$, the erroneous symbol $\hat{s}_1$ is represented by a leaf node, which can be corrected after examining at most $\left|\mathcal{O}_1^{\mathrm{FDL}}\right| - 1$ other leaf nodes, where $\left|\mathcal{O}_n^{\mathrm{FDL}}\right|$ denotes the cardinality of $\mathcal{O}_n^{\mathrm{FDL}}$. In contrast, if $\tilde{n} = N$, the erroneous symbol $\hat{s}_N$ is represented by the node closest to the root, and it is only corrected after examining the entire sub-tree having $\hat{s}_N$ as the root. Consequently, much

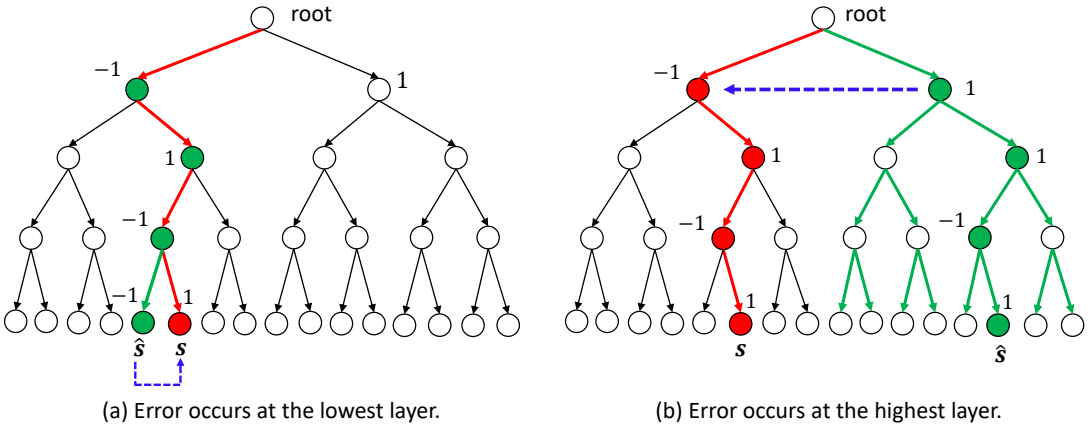


FIG. 5.2. Illustration of the effect of erroneous layer on the search efficiency of SD. The dashed arrow presents the process of exploration and correction of the erroneous symbol.

higher computational complexity is required than that required to examine only the leaf nodes, as in the case for $\tilde{n} = 1$.

According to Remark 1, the optimal solution can be found earlier if the errors exist at low layers. This can be illustrated in Fig. 5.2 for a MIMO system with $N = 4$ and QPSK signaling. We assume that the optimal solution is $\boldsymbol{s} = [-1, 1, -1, 1]^T$ and that there is only a single erroneous symbol in $\hat{\boldsymbol{s}}$. In Fig. 5.2A, $\hat{\boldsymbol{s}} = [-1, 1, -1, -1]^T$, and the error occurs at the lowest layer, i.e., $\tilde{n} = 1$, yielding $\hat{s}_1 \neq s_1$. It is observed that only one node is required to be examined to reach the optimal solution. In contrast, in Fig. 5.2B, we assume $\hat{\boldsymbol{s}} = [1, 1, -1, 1]^T$ and $\hat{s}_4 \neq s_4$, i.e., $\tilde{n} = N$. In this case, the path associated with $\hat{\boldsymbol{s}}$ is in a totally different sub-tree from that associated with $\boldsymbol{s}$. As a result, a large number of nodes are explored to correct $\hat{s}_4$ and find the optimal solution. The example in Fig. 5.2 clearly shows that the search efficiency in SD significantly depends on the erroneous layer. Motivated by this, we propose a layer-ordering scheme such that errors are more likely to occur at low layers.

In this scheme, the accuracy of the symbols in the layers of $\hat{\boldsymbol{s}}$ are evaluated. For this

purpose, we propose exploiting the difference between $\hat{\boldsymbol{s}}^{[L]}$ and $\hat{\boldsymbol{s}}$, which are the output of the last layer and the final solution of the FS-Net, respectively. We recall that $\hat{\boldsymbol{s}}^{[L]} \in \mathbb{R}^{N\times 1}$ can contain elements both inside and outside alphabet $\mathcal{A}$, as observed from step 5 in Algorithm 4 and Fig. 4.2. In contrast, $\hat{\boldsymbol{s}} = \mathcal{Q}\left(\hat{\boldsymbol{s}}^{[L]}\right) \in \mathcal{A}^N$. Let e_n denote the distance between $\hat{s}_n$ and $\hat{s}_n^{[L]}$, i.e., $e_n = \left|\hat{s}_n^{[L]} - \hat{s}_n\right|$. For QAM signals, the distance between two neighboring real symbols is two. Furthermore, from Fig. 4.2, $\hat{s}_n^{[L]} \in [-1, 1]$ for QPSK and $\hat{s}_n^{[L]} \in [-3, 3]$ for 16-QAM. Therefore, $0 \leq e_n \leq 1, n = 1, 2, \ldots, N$. It is observed that if $e_n \approx 0$, there is a high probability that the nth symbol in $\hat{\boldsymbol{s}}$ is correctly approximated by the FS-Net, i.e., $s_n = \hat{s}_n$. In contrast, if $e_n \approx 1$, there is a high probability that $\hat{s}_n$ is an erroneous estimate, i.e., $s_n \neq \hat{s}_n$. Therefore, by examining the elements of $\boldsymbol{e} = [e_1, e_2, \ldots, e_N]$, we can determine the layers with high probabilities of errors.

Based on $\hat{\boldsymbol{s}}^{[L]}$ and $\hat{\boldsymbol{s}}$, $\boldsymbol{e}$ is computed, and the layers are ordered in decreasing order of the elements of $\boldsymbol{e}$ to increase the likelihood that the errors occur at the low layers. We note that ordering the layers is equivalent to order the elements of $\boldsymbol{x}$, which requires the corresponding column ordering of $\boldsymbol{H}$. Therefore, in the proposed layer-ordering scheme, the channel columns are also ordered in the decreasing order of the elements of $\boldsymbol{e}$.

Example 1: Consider a MIMO system with $N = 8$, QPSK, and $\hat{\boldsymbol{s}}^{[L]} = [0.1, -0.65, -0.2, 0.85, 0.25, 0.9, -0.3, 1]^T$, $\hat{\boldsymbol{s}} = [1, -1, -1, 1, 1, 1, -1, 1]^T$. Then, $\boldsymbol{e} = \left|\hat{\boldsymbol{s}}^{[L]} - \hat{\boldsymbol{s}}\right| = [0.9, 0.35, 0.8, 0.15, 0.75, 0.1, 0.7, 0]^T$, which implies that the layer order should be $\{1, 3, 5, 7, 2, 4, 6, 8\}$. Consequently, the channel columns should be ordered as $\underline{\boldsymbol{H}} = [\boldsymbol{h}_1, \boldsymbol{h}_3, \boldsymbol{h}_5, \boldsymbol{h}_7, \boldsymbol{h}_2, \boldsymbol{h}_4, \boldsymbol{h}_6, \boldsymbol{h}_8]$, where $\boldsymbol{h}_n$ is the nth column of $\boldsymbol{H}$.

5.3.3. FDL-SD algorithm

Algorithm 6 FDL-SD algorithm

Input: $\boldsymbol{H}, \boldsymbol{y}$.
Output: $\hat{\boldsymbol{s}}_{SD}$.

1: Find $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$ based on Algorithm 4.
2: Obtain $\boldsymbol{e} = [e_1, e_2, \ldots, e_N]^T$, where $e_n = \left|\hat{s}_n - \hat{s}_n^{[L]}\right|$.
3: Order the channel columns in decreasing order of the elements of $\boldsymbol{e}$ to obtain $\underline{\boldsymbol{H}}$.
4: Perform the QR decomposition of $\underline{\boldsymbol{H}}$ to obtain $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{R}$.
5: Set $n = N$, $\boldsymbol{z} = \boldsymbol{Q}_1^T \boldsymbol{y}$, $d_N^2 = d^2 - \left\|\boldsymbol{Q}_2^T \boldsymbol{y}\right\|^2$.
6: Compute $z_{n|n+1}$, d_n^2, LB_n, and UB_n based on (5.3)–(5.6).
7: Obtain $\mathcal{O}_n^{\mathrm{FDL}}$ based on (5.7).
8: **if** $\mathcal{O}_n^{\mathrm{FDL}}$ is empty **then**
9: $\quad n \leftarrow n + 1$
10: **else**
11: $\quad$ Set x_n to the first element in $\mathcal{O}_n^{\mathrm{FDL}}$.
12: **end if**
13: **if** $n = 1$ **then**
14: $\quad \phi(\boldsymbol{x}) = d_N^2 - d_1^2 + (z_1 - r_{1,1}x_1)^2$
15: $\quad$ **if** $\phi(\boldsymbol{x}) \leq d_N^2$ **then**
16: $\quad\quad$ Update $d_N^2 = \phi(\boldsymbol{x})$ and $\hat{\boldsymbol{s}}_{SD} = \boldsymbol{x}$.
17: $\quad\quad$ Remove the first element of $\mathcal{O}_n^{\mathrm{FDL}}$ and go to step 8.
18: $\quad$ **end if**
19: **else**
20: $\quad$ Set $n \leftarrow n - 1$ and go to step 6.
21: **end if**

The FDL-SD algorithm is summarized in Algorithm 6, which mainly follows the conventional SD algorithm presented in [54], except for the application of candidate and layer ordering. Specifically, in step 1, the FS-Net is employed to obtain $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$, which is then used in steps 2 and 3 for layer ordering. In the remaining steps, the common search process of SD is conducted to obtain $\hat{\boldsymbol{s}}_{SD}$. Note that in step 7, all the symbols belonging to the interval $[LB_n, UB_n]$ are ordered by increasing distance from $\hat{s}_n$, as given in (5.7). Performing this operation for every layer allows the candidates to be examined by their increasing distance to the FS-Net's solution $\hat{\boldsymbol{s}}$. The remaining steps follow the well-known search procedure of SD.

Compared to the existing DL-aided SD schemes in [34, 38, 39], the proposed FDL-SD algorithm is advantageous in the following aspects:

- The application of DL in this scheme is to generate a highly reliable candidate $\hat{\boldsymbol{s}}$. We note that in this employment, the DNN, i.e., the FS-Net, can be trained without performing the conventional SD scheme, as will be further discussed in Section 5.5. In contrast, in [34,38,39], DL is applied to predict the radius, and its training labels are obtained by performing the conventional SD scheme. This requires considerable time and computational resources. For example, to train the DNN in [39] for a 16×16 MIMO system with QPSK, 100,000 samples are used, requiring performing the conventional SD 100,000 times to collect the same number of desired radii for training, whereas that number required in [34] for a 10×10 MIMO system with 16-QAM is 360,000. This computational burden in the training phase of the existing DL-aided SD schemes is non-negligible, even for offline processing.

- The proposed scheme does not require optimizing the initial radius, as in [34,38,39], because the initial sphere is predetermined based on $\hat{\boldsymbol{s}}$. Note that in the conventional SD, if the radius is initialized to a small value, it is possible that there will be no point inside the sphere. In this case, the search needs to restart with a larger radius, resulting in redundant complexity. In contrast, in the FDL-SD scheme, starting with $\hat{\boldsymbol{s}}$ guarantees that there is always at least one point inside the sphere, which is nothing but $\hat{\boldsymbol{s}}$. Furthermore, because $\hat{\boldsymbol{s}}$ has high accuracy, the number of points inside the sphere is typically small.

- In the proposed FDL-SD, the search efficiency is improved, thus providing significant complexity reduction. Despite that, the ordering schemes in the FDL-SD do not affect the radius or terminate the search early, as in [34, 38, 39]. Therefore, the BER

performance of the conventional SD is totally preserved in the proposed FDL-SD scheme. We will further justify this with the simulation results in Section 5.5.

5.4. Proposed FDL-KSD scheme

It is intuitive from the tree-like model shown in Fig. 5.1B that there are $|\mathcal{A}|^N$ complete paths representing all the possible candidates for the optimal solution, where $|\mathcal{A}| = 2$ and $N = 4$ for the example in Fig. 5.1B. In a large MIMO system with a high-order modulation scheme, i.e., when N and $|\mathcal{A}|$ are large, the number of paths becomes very large. Therefore, in the KSD, instead of examining all the available paths, only the K best paths are selected in each layer for further extension to the lower layer, while the others are pruned early to reduce complexity. For the selection of the K best paths, each path is evaluated based on its metric. Specifically, in layer n, if the kth path extends to a node x_i, its metric is given by

$$\phi_n^{(k,i)} = \phi_{n+1}^{(k)} + \left(z_n - \sum_{i=n}^{N} r_{n,i} x_i \right)^2, \tag{5.9}$$

with $\phi_{N+1}^{(k)} = 0, \forall k$. Then, only a subset of K paths with the smallest metrics $\{\phi_n^{(1)}, \ldots, \phi_n^{(K)}\}$ are selected for further extension. In the lowest layer, the best path with the smallest metric is concluded to be the final solution. In this study, to further optimize the KSD scheme in terms of both complexity and performance, we propose the FDL-KSD scheme with early rejection and layer ordering, which is presented in the following subsection.

5.4.1. Basic ideas: early rejection and layer ordering

Early rejection: The idea of early rejection is that, given a candidate $\hat{\boldsymbol{s}}$, a candidate that is worse than $\hat{\boldsymbol{s}}$ cannot be the optimal solution, and it can be rejected early from the examination process. This definitely results in complexity reduction without any performance loss. To apply this idea to the KSD, among the K chosen paths in each layer of the KSD scheme, the paths with metrics larger than $\phi(\hat{\boldsymbol{s}})$ are pruned early. It is possible that all the K paths are pruned in a layer if all of them are worse than $\hat{\boldsymbol{s}}$. In this case, there is no path for further extension. Hence, the examination process is terminated early, and $\hat{\boldsymbol{s}}$ is concluded to be the final solution. It is observed that in this early rejection approach, the final solution is the best one between that attained by the conventional KSD and the FS-Net-based solution. Therefore, besides providing complexity reduction, this scheme also attains performance improvement w.r.t. the conventional KSD.

Layer ordering: One potential problem of the KSD is that the optimal solution can be rejected before the lowest layer is reached, causing its performance loss w.r.t. the sequential SD. An approach to mitigate the unexpected early rejection of the optimal solution is to apply the layer-ordering scheme proposed in Section 5.3. Specifically, it is observed from (5.9) that if the elements of a candidate $\boldsymbol{x}$ are ordered such that the ones in higher layers are more reliable than those in lower layers, then the best path is more likely to have small metrics at high layers. As a result, the chance that it is early pruned is reduced. Therefore, we propose applying the layer-ordering scheme proposed in Section 5.3 to the FDL-KSD scheme for performance improvement.

5.4.2. FDL-KSD algorithm

Algorithm 7 FDL-KSD algorithm

Input: $\boldsymbol{H}, \boldsymbol{y}, K$.
Output: $\hat{\boldsymbol{s}}_{KSD}$.

1: Find $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}^{[L]}$ based on Algorithm 4.
2: Obtain $\boldsymbol{e} = [e_1, e_2, \ldots, e_N]^T$, where $e_n = \left|\hat{s}_n - \hat{s}_n^{[L]}\right|$.
3: Order the channel columns in decreasing order of the elements of $\boldsymbol{e}$ to obtain $\underline{\boldsymbol{H}}$.
4: Perform QR decomposition of $\underline{\boldsymbol{H}}$ to obtain $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, and $\boldsymbol{R}$.
5: $\boldsymbol{z} = \boldsymbol{Q}_1^T \boldsymbol{y}$
6: **for** $n = N \rightarrow 1$ **do**
7: Determine the K best paths associated with the K smallest metrics $\{\phi_n^{(1)}, \ldots, \phi_n^{(K)}\}$.
8: Prune the paths that have metrics larger than $\phi(\hat{\boldsymbol{s}})$ early.
9: **if** all paths have been pruned **then**
10: Terminate the search early.
11: **end if**
12: Save the survival paths for further extension.
13: **end for**
14: **if** there is no survival path **then**
15: Set $\hat{\boldsymbol{s}}_{KSD} = \hat{\boldsymbol{s}}$.
16: **else**
17: Set $\hat{\boldsymbol{s}}_{KSD}$ to the survival candidate corresponding to the path with the smallest metric.
18: **end if**

The proposed FDL-KSD scheme is summarized in Algorithm 7. In steps 1–3, layer ordering is performed. The K best paths are selected in step 7, and a subset of them with metrics larger than $\phi(\hat{\boldsymbol{s}})$ are pruned early in step 8. In the case where all the paths are pruned, the path examination and extension process is terminated early in step 10, and the FS-Net-based solution $\hat{\boldsymbol{s}}$ is concluded to be the final solution, as shown in step 15. In contrast, if early termination does not occur, the search continues until the lowest layer is reached, at which the final solution $\hat{\boldsymbol{s}}_{KSD}$ is set to be the best candidate among the surviving ones, as in step 17.

We discuss the properties of K, i.e., the number of survival paths in the proposed FDL-KSD scheme. It can be seen that in the proposed scheme, the number of actual survival paths is dynamic, whereas it is fixed to K in the conventional KSD scheme. Letting K_n

be the number of survival paths in the nth layer, we have $K_n \leq K, \forall n$, which is clear from step 8 of Algorithm 7. Furthermore, it is observed in (5.9) that the paths' metrics increase with n. As a result, more paths have metrics exceeding $\phi(\hat{\boldsymbol{s}})$ as n increases. Consequently, the number of survival paths becomes smaller as the search goes downward to lower layers, i.e., $K_1 \geq K_2 \geq \ldots \geq K_N$.

These properties make the design of the FDL-KSD scheme much easier than that of the conventional KSD scheme. We first note one challenge in the conventional KSD, which is to choose the optimal value for K. Specifically, if a large K is set, many candidates are examined, resulting in high complexity. In this case, the complexity reduction of KSD w.r.t. the conventional SD is not guaranteed. In contrast, a small K leads to significant performance loss because there is a high probability that the optimal path is pruned before the lowest layer is reached. It is possible to use dynamic K, i.e., to set different values of K for different layers. However, optimizing multiple values of $\{K_1, K_2, \ldots, K_N\}$ becomes problematic, as N is large in large MIMO systems. In the proposed FDL-KSD scheme, K_n is already dynamic. Furthermore, because K_n is adjusted in step 8 of Algorithm 7, we only need to set K to a sufficiently large value to guarantee near-optimal performance, and unpromising paths are automatically rejected by the FDL-KSD scheme.

5.5. Simulation results

In this section, we numerically evaluate the BER performance and computational complexities of the proposed FDL-SD and FDL-KSD schemes. The computational complexity of an algorithm is calculated as the total number of additions and multiplications. In our simulations, each channel coefficient is assumed to be an i.i.d. zero-mean complex Gaussian

TABLE 5.1. Architectures and complexities of the DNNs used in the MR-DL-SD, DDP-SD, and the proposed FDL-SD schemes for a 16×16 MIMO system with QPSK.

DNNs	No. input nodes	No. hidden nodes $\times$ No. hidden layers	No. output nodes	Complexity (operations)
FC-DNN in the MR-DL-SD	544	128×1	4	140288
FC-DNN in the DPP-SD	34	40×1	4	3040
FS-Net in the FDL-SD, FDL-KSD, and DL-TS	64	64×10	32	88608

random variable with a variance of $1/2$ per dimension. SNR is defined as the ratio of the average transmit power to the noise power, i.e., $\text{SNR} = N_t\sigma_t^2/\sigma_n^2$.

5.5.1. Training DNNs

The hardware and software used for implementing and training the DNNs are as follows. The FS-Net is implemented by using Python with the TensorFlow library [51]. In contrast, the FC-DNNs in the MR-DL-SD and DPP-SD are implemented using the DL Toolbox of MATLAB 2019a, as done in [34] and [39]. All the considered DNNs, i.e., the FS-Net and FC-DNNs, are trained by the Adam optimizer [48–50] with decaying and starting learning rates of 0.97 and 0.001, respectively. The FC-DNNs used for the MR-DL-SD and DPP-SD are trained for 100,000 samples, as in [39]. In contrast, we train the FS-Net for 35,000 iterations with batch sizes of 2,000 samples, as in [37]. For each sample, $\boldsymbol{s}, \boldsymbol{H}$, and $\boldsymbol{y}$ are independently generated from (1.2).

As discussed in Section 5.1, one of the significant differences between the proposed and existing DL-aided SD schemes lies in the training phase. In the existing DL-aided SD schemes, including the SR-DL-SD, MR-DL-SD, and DPP-SD, the FC-DNNs are trained

with the following loss function [34, 39]:

$$\mathcal{L}\left(\boldsymbol{d}^{(i)}, \hat{\boldsymbol{d}}^{(i)}\right) = \frac{1}{D}\sum_{i=1}^{D}\left\|\boldsymbol{d}^{(i)} - \hat{\boldsymbol{d}}^{(i)}\right\|^2,$$

where D is the number of training data samples, and $\boldsymbol{d}^{(i)}$ and $\hat{\boldsymbol{d}}^{(i)}$ are the label and output vectors of the DNNs, which represent the radii associated with the ith data sample. The training labels, i.e., $\{\boldsymbol{d}^{(1)}, \ldots, \boldsymbol{d}^{(D)}\}$, are obtained by performing the conventional SD scheme for D times, where $D = \{100{,}000, 360{,}000\}$ for the DPP-SD and MR-DL-SD schemes, respectively [34, 39]. It is well known that the conventional SD is computationally prohibitive for large MIMO systems. Therefore, huge amounts of computational resources and time are required to collect a huge training data set in the existing DL-aided SD schemes.

In contrast, in the proposed application of DL to SD, the FS-Net is employed to generate $\hat{\boldsymbol{s}}$. The FS-Net is trained with the loss function [37]

$$\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \sum_{l=1}^{L}\log(l)\left[\left\|\boldsymbol{s} - \hat{\boldsymbol{s}}^{[l]}\right\|^2 + \xi r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s})\right], \tag{5.10}$$

where $\hat{\boldsymbol{s}}^{[l]}$ and $\boldsymbol{s}$ are the output of the lth layer and the desired transmitted signal vector, respectively, and $r(\hat{\boldsymbol{s}}^{[l]}, \boldsymbol{s}) = 1 - \frac{\left|\boldsymbol{s}^T\hat{\boldsymbol{s}}^{[l]}\right|}{\|\boldsymbol{s}\|\|\hat{\boldsymbol{s}}^{[l]}\|}$. As the training labels $\boldsymbol{s}$ of the FS-Net are generated randomly, the conventional SD does not need to be performed to generate the training labels as done in the existing DL-aided SD schemes.

5.5.2. BER performance and computational complexity of the proposed FDL-SD algorithm

In this section, we show the BER performance and computational complexity of the proposed FDL-SD scheme, which are compared to those of two conventional SD schemes, FP-SD and SE-SD, and the existing DL-aided SD schemes, including the MR-DL-SD [34] and DPP-SD [39]. Furthermore, we consider the DL-aided tabu search (DL-TS) scheme [37], which also leverages the FS-Net for complexity reduction. We will later show that, although the conventional SD requires much higher complexity than the TS scheme [35], the application of DL makes the FDL-SD more complexity-efficient than DL-TS. For the proposed FDL-SD, we also show its BER performance and computational complexity when only the candidate ordering is applied; with this data, we can compare the efficiency of order $\mathcal{O}_n^{\mathrm{FDL}}$ in (5.7) proposed for the FDL-SD scheme and $\mathcal{O}_n^{\mathrm{SE}}$ in (5.8) employed in the conventional SE-SD scheme.

Furthermore, we note that the structures and complexities of the DNNs employed in the compared schemes are different, as illustrated in Table 5.1 for a 16×16 MIMO system with QPSK. The DNNs used in the MR-DL-SD and DPP-SD schemes have well-known fully-connected architectures, and it is clear that their complexities are twice the total number of connections in the network. Here, the factor of two is to account for one multiplication with the weight and one addition with the bias associated with each connection. In contrast, the complexity of the FS-Net is computed based on (4.15). In our simulation results, the overall complexity of each considered scheme is computed as the sum of the complexity required in the DNNs, presented in Table 5.1, and that required to perform the algorithms themselves. The simulation parameters for the MR-DL-SD, DPP-SD, and DL-TS schemes are listed in Table 5.2, which are set based on the corresponding prior works. It is observed

TABLE 5.2. Simulation parameters for the MR-DL-SD, DPP-SD, and DL-TS schemes, where λ_1, λ_2 are the optimized design parameters of the DPP-SD scheme, $\mathcal{I}$ is the maximum number of search iterations, μ is an optimized design parameter, and ϵ is the cutoff factor for early termination in the DL-TS scheme.

Schemes	Parameters
MR-DL-SD	Number of predicted radii: 4
DPP-SD	$\lambda_1 = \{1.1, 1.2 \ldots, 1.6\}$ for SNR $= \{2, 4, \ldots, 12\}$, respectively, $\lambda_2 = \lambda_1 + 0.1$
DL-TS	$\mathcal{I} = \{400, 700\}, \mu = \{7, 8\}$ for 16×16 MIMO and 24×24 MIMO systems, respectively, and $\epsilon = 0.4$

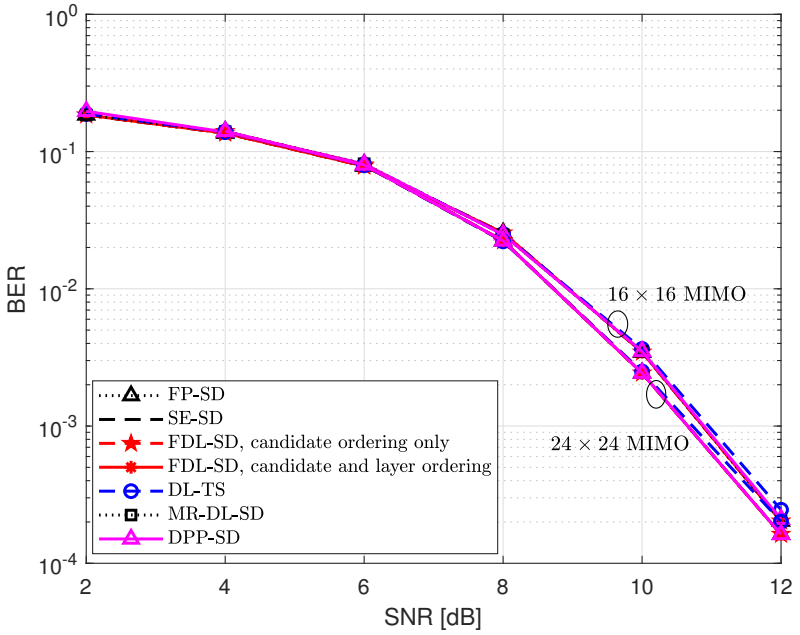


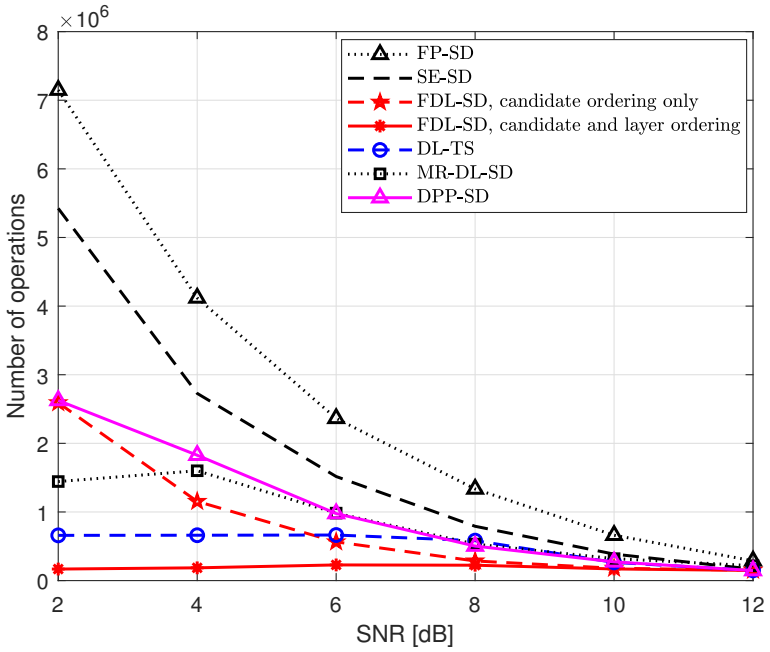
FIG. 5.3. BER performance of the proposed FDL-SD scheme compared to those of the conventional FP-SD, SE-SD, MR-DL-SD, DPP-SD, and DL-TS for $(N_t \times N_r) = (16 \times 16)$, $L = 10$ and $(N_t \times N_r) = (24 \times 24)$, $L = 12$ with QPSK.

that an advantage of the proposed FDL-SD scheme is that it does not require optimizing any design parameters, as done in the DL-aided detection algorithms in Table 5.2.

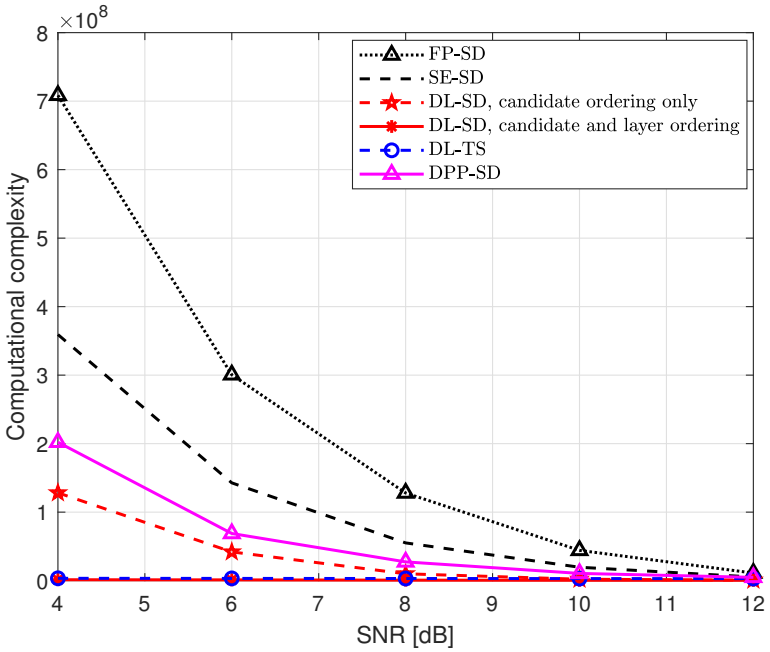
In Fig. 5.3, we show the BER performance of the schemes listed earlier for a 16×16 MIMO system with $L = 10$ and a 24×24 MIMO system with $L = 12$, both with QPSK. We note that the simulation for the MR-DL-SD scheme is omitted for the 24×24 MIMO system because it takes an extremely long time to collect the desired radii to

train its associated DNN. It is observed from Fig. 5.3 that all the compared schemes have approximately the same BER performance in both considered MIMO systems. In particular, the proposed FDL-SD schemes, including that with candidate ordering only and that with both candidate and layer ordering, totally preserve the performance of the conventional FP-SD and SE-SD. In contrast, the MR-DL-SD, DPP-SD, and DL-TS schemes have marginal performance loss w.r.t. the conventional SD schemes, which agrees with the results in [34, 37, 39].

In Fig. 5.4, we compare the proposed FDL-SD scheme to the conventional FP-SD, SE-SD, MR-DL-SD, DPP-SD, and DL-TS schemes in terms of computational complexity. To ensure that the compared schemes have approximately the same BER performance, the simulation parameters in Fig. 5.4 are assumed to be the same as those in Fig. 5.3. In Fig. 5.4, the complexity reduction gains of the considered schemes are difficult to compare at high SNRs. Therefore, we show their complexity ratios w.r.t. the complexity of the conventional FP-SD in Fig. 5.5. In other words, the complexity of all schemes are normalized by that of the FP-SD. From Figs. 5.4 and 5.5, the following observations are noted:

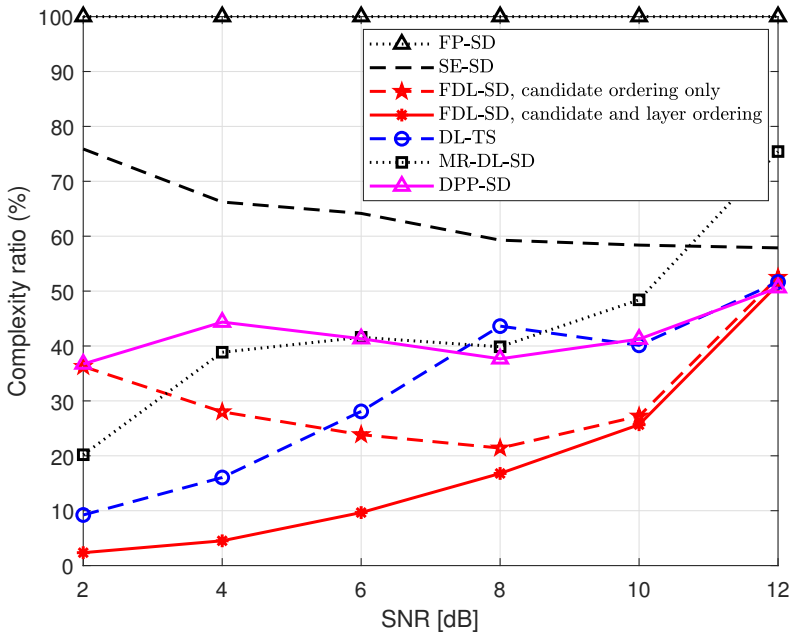


(A) 16×16 MIMO

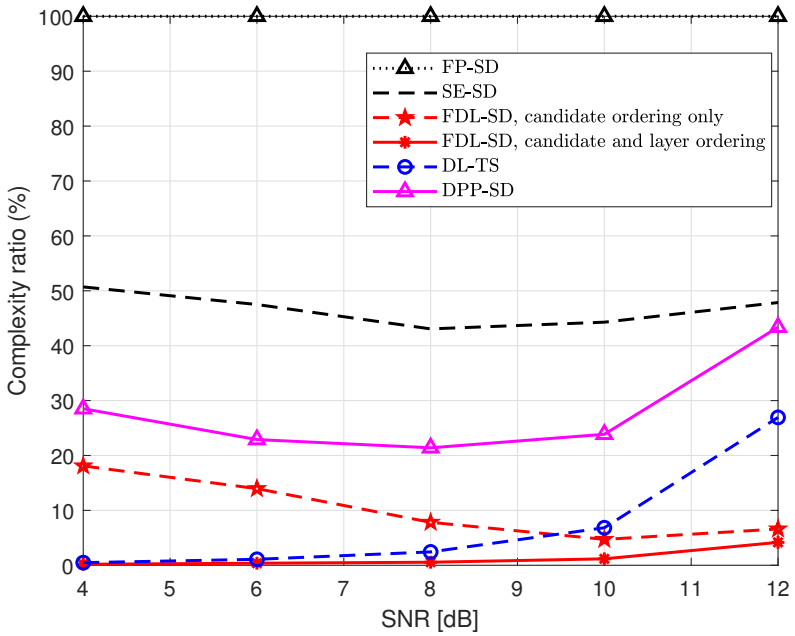


(B) 24×24 MIMO

FIG. 5.4. Complexity of the proposed FDL-SD scheme compared to those of the conventional FP-SD, SE-SD, MR-DL-SD, DPP-SD, and DL-TS for 16×16 MIMO with $L = 10$, and 24×24 MIMO with $L = 12$, both with QPSK.



(A) 16×16 MIMO



(B) 24×24 MIMO

FIG. 5.5. Complexity ratio of the proposed FDL-SD scheme compared to those of the conventional FP-SD, SE-SD, MR-DL-SD, DPP-SD, and DL-TS for 16×16 MIMO with $L = 10$, and 24×24 MIMO with $L = 12$, both with QPSK.

- It is clear from Fig. 5.4 that the complexities of the conventional FP-SD, SE-SD, and the existing DL-aided SD schemes, including the MR-DL-SD and DPP-SD, significantly depend on SNRs. In contrast, that of the proposed FDL-SD scheme is relatively stable with SNRs.

- Compared to the conventional FP-SD, the SE-SD and the proposed FDL-SD scheme with candidate ordering only are similar in the sense that symbols are ordered in each layer based on $\mathcal{O}_n^{\mathrm{FDL}}$ and $\mathcal{O}_n^{\mathrm{SE}}$, respectively. However, it can be clearly seen in Fig. 5.5 that the order $\mathcal{O}_n^{\mathrm{FDL}}$ obtained based on the FS-Net's output is considerably better than the $\mathcal{O}_n^{\mathrm{SE}}$ used in the conventional SE-SD. Specifically, in the 24×24 MIMO system, the proposed FDL-SD with candidate ordering based on $\mathcal{O}_n^{\mathrm{FDL}}$ achieves $80\% - 95\%$ complexity reduction w.r.t. the conventional FP-SD, while that achieved by the SE-SD with $\mathcal{O}_n^{\mathrm{SE}}$ is only around 50%, as seen in Fig. 5.5B.

- In Fig. 5.5, among the improved SD schemes, the proposed FDL-SD achieves the most significant complexity reduction w.r.t. the conventional FP-SD scheme. Specifically, in the 16×16 MIMO system, for SNR $\leq$ 6 dB, the complexity reduction ratios of the proposed FDL-SD are higher than 90%, while those of the MR-DL-SD and DPP-SD are only around 60%. At SNR = 12 dB, the DPP-SD and FDL-SD have approximately the same complexity, which is much lower than that of the MR-DL-SD scheme. This is due to the high complexity required for the DNN used in the MR-DL-SD scheme, as shown in Table 5.1. In the 24×24 MIMO system, the complexity reduction ratio of the FDL-SD with both candidate and layer ordering is $95\% - 98\%$, which is much higher than $55\% - 80\%$ for the DPP-SD scheme.

- Furthermore, by comparing the complexity ratios of the FDL-SD scheme in Figs.

5.5A and 5.5B, it can be observed that this scheme achieves more significant complexity reduction in a larger MIMO system. Specifically, in the 16×16 MIMO system, its complexity reduction ratio w.r.t. the conventional FP-SD is only around $50\% - 97\%$. In contrast, that in the 24×24 MIMO system is $95\% - 98\%$. The reason for this improvement is that, in large MIMO systems, the complexity of the SD algorithm significantly dominates that of the FS-Net and becomes almost the same as the overall complexity. Therefore, the complexity required in the FS-Net has almost no effect on the complexity of the FDL-SD scheme. This observation demonstrates that the proposed FDL-SD scheme is suitable for large MIMO systems.

- Notably, the proposed FDL-SD has considerably lower complexity than the DL-TS scheme although the conventional SD requires higher complexity than the TS detector [35,36]. This verifies that the application of DL makes SD a more computationally efficient detection scheme than the TS.

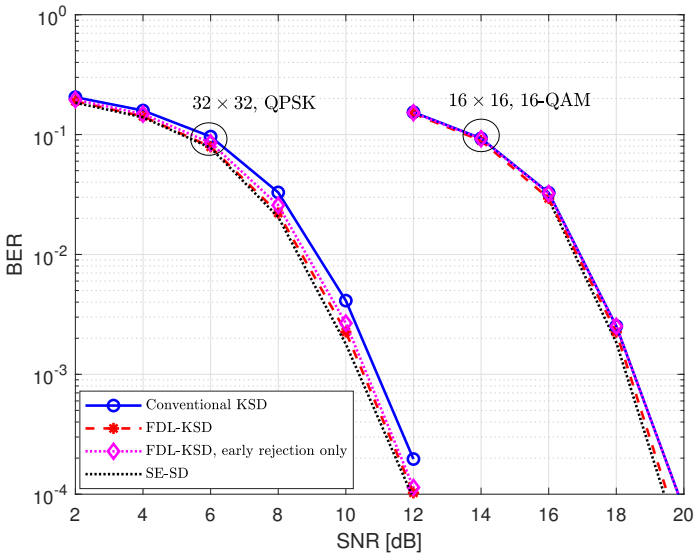
In summary, it is clear from Figs. 5.3–5.5 that the proposed FDL-SD scheme has no performance loss w.r.t. the conventional SD, whereas it attains the most significant complexity reduction among the compared schemes.

5.5.3. BER performance and computational complexity of the proposed FDL-KSD scheme

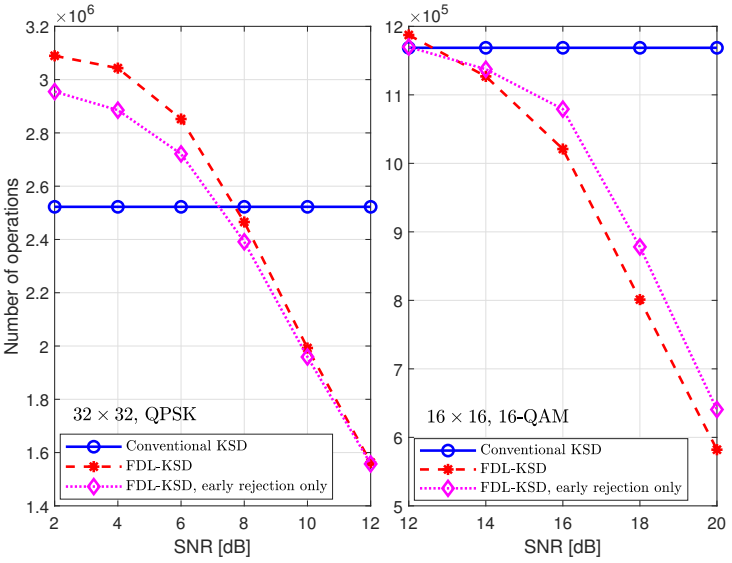
In Fig. 5.6, we show the BER performance and complexity of the proposed FDL-KSD and the conventional KSD schemes for a 32×32 MIMO system with QPSK and a 16×16 MIMO system with 16-QAM, where $L = \{15, 30\}$ are set, respectively. In both systems, $K = 256$ is chosen such that the considered KSD schemes nearly achieve the performance of

the SE-SD scheme. We note that no existing work in the literature considers the application of DL to KSD. Therefore, we only compare the performance and complexity of the proposed FDL-KSD to those of the conventional KSD. Furthermore, we also show the performance and complexity of the proposed FDL-KSD with early rejection only, to demonstrate that this early rejection scheme attains not only performance improvement, but also complexity reduction.

In Fig. 5.6, it is observed that unlike the conventional KSD scheme, whose complexity is fixed with SNRs, the proposed FDL-KSD scheme has the complexity decreasing significantly with SNRs. Specifically, the complexities of the FDL-KSD scheme in both considered systems are reduced by approximately half as the SNR increases from low to high. Moreover, it is clear that at moderate and high SNRs, the proposed FDL-KSD scheme achieves better performance with considerably lower complexity than the conventional KSD scheme. In particular, the early rejection can achieve improved performance and reduced complexity w.r.t. the conventional KSD. The additional application of candidate ordering results in further performance improvement of the FDL-KSD algorithm, as clearly seen in Fig. 5.6A. Specifically, in both considered systems, a performance improvement of 0.5 dB in SNR is achieved. At the same time, the complexity reduction of 38.3% and 50.2% w.r.t. the conventional KSD is attained at SNR = 12 dB and 20 dB for the 32×32 MIMO system with QPSK and the 16×16 MIMO system with 16-QAM, respectively. We note that the proposed FDL-KSD has higher or comparable complexity w.r.t. the conventional KSD at low SNRs because the complexity required for the FS-Net is included.

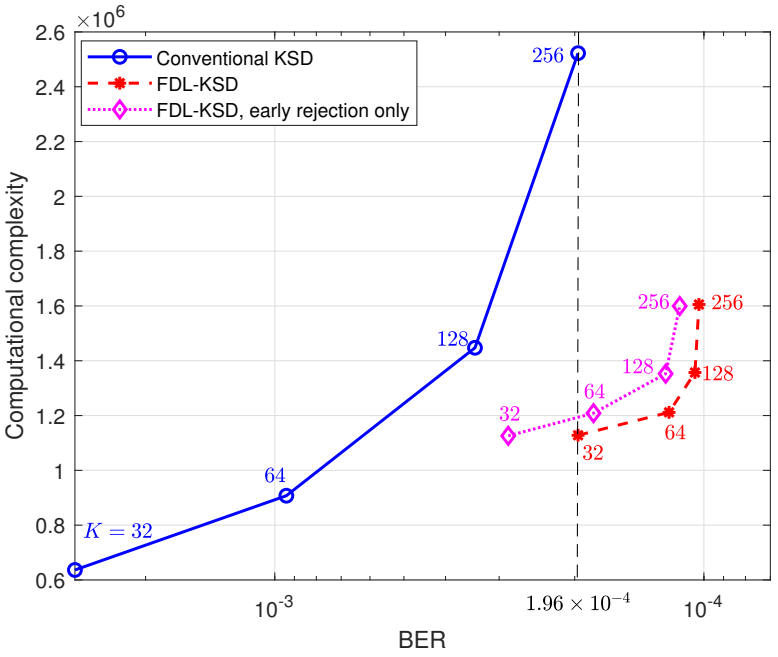


(A) BER performance

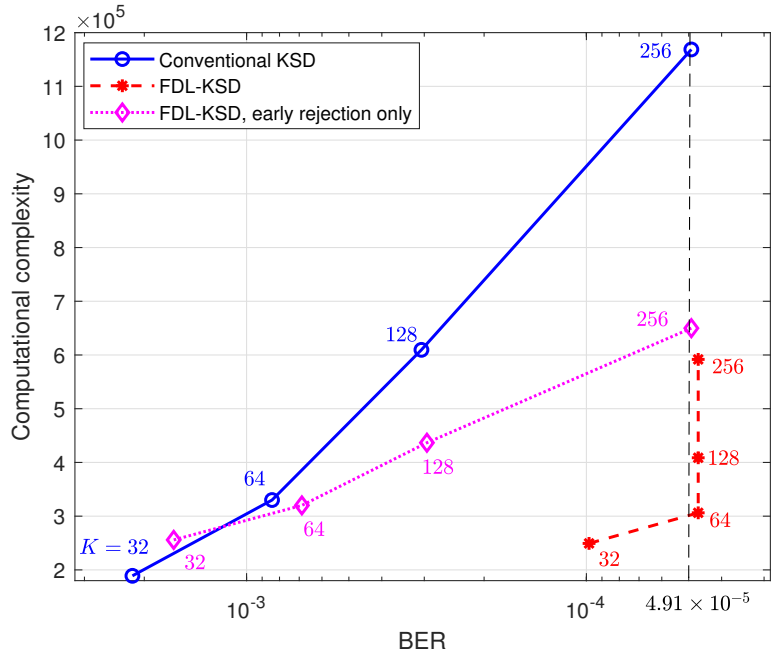


(B) Computational complexity

FIG. 5.6. BER performance and computational complexity of the proposed FDL-KSD scheme compared to that of the conventional KSD. Simulation parameters are $(N_t \times N_r) = (32 \times 32)$, $L = 15$ for QPSK, $(N_t \times N_r) = (16 \times 16)$, $L = 30$ for 16-QAM, and $K = 256$.



(A) 32×32 MIMO, $L = 15$, SNR $= 12$ dB with QPSK



(B) 16×16 MIMO, $L = 30$, SNR $= 20$ dB with 16-QAM

FIG. 5.7. Tradeoff between BER performance and computational complexity of the proposed FDL-KSD scheme compared to that of the conventional KSD for $K = \{32, 64, 128, 256\}$.

In Fig. 5.7, we show the improvement in the performance–complexity tradeoff of the proposed FDL-KSD scheme w.r.t. the conventional KSD scheme for a 32×32 MIMO system with QPSK and SNR = 12 dB and a 16×16 MIMO system with 16-QAM and SNR = 20 dB. Various values for K are considered, including $K = \{32, 64, 128, 256\}$. We make the following observations:

- First, it is clear that the proposed FDL-KSD scheme not only achieves better BER performance but also requires much lower complexity than the conventional KSD scheme. For example, to attain a BER of 1.96×10^{-4} in the 32×32 MIMO system with QPSK, the conventional KSD scheme requires $K = 256$, whereas only $K < 64$ is sufficient for the proposed FDL-KSD scheme with early rejection only, corresponding to a complexity reduction ratio of 52.5%, and only $K = 32$ is required for the FDL-KSD scheme with both early rejection and candidate ordering, resulting in 55.6% complexity reduction.

- Second, the complexity reduction is more significant as K increases. This is because in the proposed FDL-KSD scheme, the number of actual survival nodes is not K, but $K_n \leq K, \forall n$, as discussed in Section 5.4.

- Moreover, the performance–complexity tradeoff of the conventional KSD scheme significantly depends on K, as discussed in Section 5.4. In Fig. 5.7, its BER performance can be improved dramatically as K increases, which, however, causes considerably high complexity. In contrast, the performance–complexity tradeoff of the proposed FDL-KSD scheme is comparatively stable with K. For example, its BER performance in the 32×32 MIMO system with QPSK is approximately the same for $K = 128$ and $K = 256$, and its complexity increases relatively slowly as K increases. In contrast, in the 16×16 MIMO system with 16-QAM, $K = 64$ is sufficient to achieve a BER of

4.68×10^{-5}, and a further increase of K to $\{128, 256\}$ does not result in performance improvement. In this case, we can conclude that $K = 64$ is optimal for the FDL-KSD scheme.

5.6. Conclusion

In this chapter, we have presented a novel application of DL to both the conventional SD and KSD, resulting in the FDL-SD and FDL-KSD schemes, respectively. The main idea is to leverage the FS-Net to generate a highly reliable initial solution with low complexity. The initial solution determined by the FS-Net is exploited for candidate and layer ordering in the FDL-SD scheme, and for early rejection and layer ordering in the FDL-KSD scheme. Unlike the existing DL-aided SD schemes, the proposed application of DL to the SD schemes does not require performing the conventional SD schemes to generate the training data. Therefore, the employed DNN, i.e., FS-Net, can be trained with significantly less time and computational resources than those required in existing works. Our simulation results justify the performance and complexity-reduction gains of the proposed schemes. Specifically, the FDL-SD scheme achieves remarkable complexity reduction, which exceeds 90%, without any performance loss. Moreover, the proposed FDL-KSD scheme attains a dramatically improved performance–complexity tradeoff.

6. Conclusion

In this thesis, we propose five novel near-optimal reduced-complexity detection algorithms, namely, QR-TS, NG-TS, DL-TS, FDL-SD, FDL-KSD, for large MIMO systems. The proposed QR-TS scheme allows for finding the best neighbor without computing all the neighbors' metrics by early-rejecting unpromising neighbors. By contrast, the NG-TS algorithm divides the neighbors into groups and finds the best neighbor by using a simplified cost function. To further optimize the QR-TS and NG-TS algorithms, novel ordering schemes are proposed. Simulation results show that QR-TS achieves a complexity reduction of approximately 60% – 90% w.r.t. the conventional TS scheme, while that attained by the NG-TS is up to 85%, without any performance loss.

In the proposed DL-aided detection schemes, namely, DL-TS, FDL-SD, and FDL-KSD, the proposed FS-Net is incorporated with the TS, SD, and KSD schemes, respectively. Specifically, in the DL-TS scheme, the highly reliable initial solution is generated by the FS-Net, and an adaptive ET algorithm and a modified searching process are performed so that the optimal solution can be reached earlier. The FS-Net is also leveraged in the proposed FDL-SD and FDL-KSD schemes to generate a promising initial candidate so that the search in the SD and KSD can be accelerated in conjunction with candidate/layer ordering and early rejection. The simulation results show that a complexity reduction ratio of more than 90% w.r.t. the conventional/existing algorithms is attained by the proposed schemes with no/marginal performance loss.

A. Proof of Lemma 2.1

The expectations of layer metrics, i.e., $\mathbb{E}\left\{\left|z_n + r_{n,d}\delta_d\right|^2\right\}$, can be expressed as

$$\mathbb{E}\left\{\left|z_n + r_{n,d}\delta_d\right|^2\right\} = \mathbb{E}\left\{\left|z_n\right|^2 + \left|\delta_d\right|^2 \left|r_{n,d}\right|^2 + 2z_n r_{n,d}\delta_d\right\}. \tag{A.1}$$

For each modulation scheme, $\left|\delta_d\right|^2$ is a constant. For example, in QPSK/16-QAM, $\delta_d = \pm 2$, and hence $\left|\delta_d\right|^2 = 4$. Therefore, we drop the index d in $\left|\delta_d\right|^2$ for simplicity, which yields

$$\begin{aligned}\mathbb{E}\left\{\left|z_n + r_{n,d}\delta_d\right|^2\right\} = \mathbb{E}\left\{\left|z_n\right|^2\right\} + \left|r_{n,d}\right|^2 \delta^2 \\ + 2r_{n,d}\mathbb{E}\left\{z_n\delta_d\right\}.\end{aligned} \tag{A.2}$$

We first evaluate $\mathbb{E}\left\{\left|z_n\right|^2\right\}$. By denoting the nth rows of $\boldsymbol{Q}^T$ and $\boldsymbol{R}$ as $\boldsymbol{q}_n^T$ and $\boldsymbol{p}_n^T$, respectively, we obtain

$$\begin{aligned}z_n &= \boldsymbol{q}_n^T\boldsymbol{y} - \boldsymbol{p}_n^T\boldsymbol{c} = \boldsymbol{q}_n^T(\boldsymbol{H}\,\boldsymbol{s} + \boldsymbol{v}) - \boldsymbol{p}_n^T\boldsymbol{c} \\ &= \boldsymbol{p}_n^T\left(\boldsymbol{s} - \boldsymbol{c}\right) + \boldsymbol{q}_n^T\boldsymbol{v},\end{aligned} \tag{A.3}$$

which leads to

$$\begin{aligned}\left|z_n\right|^2 &= \left|\boldsymbol{p}_n^T\left(\boldsymbol{s} - \boldsymbol{c}\right) + \boldsymbol{q}_n^T\boldsymbol{v}\right|^2 \\ &= \left|\boldsymbol{p}_n^T\left(\boldsymbol{s} - \boldsymbol{c}\right)\right|^2 + \left|\boldsymbol{q}_n^T\boldsymbol{v}\right|^2 + 2\boldsymbol{p}_n^T\left(\boldsymbol{s} - \boldsymbol{c}\right)\boldsymbol{q}_n^T\boldsymbol{v}.\end{aligned} \tag{A.4}$$

Because $\boldsymbol{v}$ is a zero-mean random vector and is independent of $\boldsymbol{s}$ and $\boldsymbol{c}$, the last term in (A.4) has zero mean, i.e., $\mathbb{E}\left\{\boldsymbol{p}_n^T\left(\boldsymbol{s}-\boldsymbol{c}\right)\boldsymbol{q}_n^T\boldsymbol{v}\right\}=0$. Hence, we have

$$\begin{aligned}
\mathbb{E}\left\{|z_n|^2\right\} &= \mathbb{E}\left\{\left|\boldsymbol{p}_n^T\left(\boldsymbol{s}-\boldsymbol{c}\right)\right|^2\right\}+\mathbb{E}\left\{\left|\boldsymbol{q}_n^T\boldsymbol{v}\right|^2\right\} \\
&= \mathbb{E}\left\{\left|\sum_{i=1}^{N} r_{n,i}(s_i-c_i)\right|^2\right\}+\mathbb{E}\left\{\left|\sum_{i=1}^{N} q_{n,i}v_i\right|^2\right\} \\
&= \mathbb{E}\left\{\sum_{i=1}^{N}\left|r_{n,i}(s_i-c_i)\right|^2\right\} \\
&\quad + \underbrace{2\,\mathbb{E}\left\{\sum_{i=1}^{N}\sum_{j>i}^{N} r_{n,i}(s_i-c_i)r_{n,j}(s_j-c_j)\right\}}_{=0} \qquad \text{(A.5)} \\
&\quad + \mathbb{E}\left\{\sum_{i=1}^{N}\left|q_{n,i}\right|^2\left|v_i\right|^2\right\} \\
&\quad + \underbrace{2\,\mathbb{E}\left\{\sum_{i=1}^{N}\sum_{j>i}^{N} q_{n,i}v_i q_{n,j}v_j\right\}}_{=0} \qquad \text{(A.6)} \\
&= \sum_{i=1}^{N}\left|r_{n,i}\right|^2\mathbb{E}\left\{\left|s_i\right|^2\right\}+\sum_{i=1}^{N}\left|r_{n,i}\right|^2\mathbb{E}\left\{\left|c_i\right|^2\right\} \\
&\quad - 2\sum_{i=1}^{N}\left|r_{n,i}\right|^2\mathbb{E}\left\{s_i c_i\right\}+\sum_{i=1}^{N}\left|q_{n,i}\right|^2\mathbb{E}\left\{\left|v_i\right|^2\right\}. \qquad \text{(A.7)}
\end{aligned}$$

In (A.5), we use $\mathbb{E}\left\{\sum_{i=1}^{N}\sum_{j>i}^{N} r_{n,i}(s_i-c_i)r_{n,j}(s_j-c_j)\right\}=0$ because s_i-c_i and s_j-c_j are independent zero-mean random variables when $i\neq j$. Similarly, in (A.6), we have $\mathbb{E}\left\{\sum_{i=1}^{N}\sum_{j>i}^{N} q_{n,i}v_i q_{n,j}v_j\right\}=0$ because the elements of $\boldsymbol{v}$ are independent and have $\mathcal{CN}(0,\sigma_v^2)$ distribution. Because s_i and c_i are drawn from the alphabet $\mathcal{A}$, we have $\mathbb{E}\left\{\left|c_i\right|^2\right\}=\mathbb{E}\left\{\left|s_i\right|^2\right\}=\sigma_t^2$, where σ_t^2 is the average symbol power.

By letting $\varepsilon = \mathbb{E}\{s_i c_i\}, i = 1, 2, \ldots, N$, (A.7) can be rewritten as

$$\mathbb{E}\left\{|z_n|^2\right\} = 2\left(\sigma_t^2 - \varepsilon\right)\sum_{i=1}^{N}|r_{n,i}|^2 + \sigma_v^2\sum_{i=1}^{N}|q_{n,i}|^2 . \tag{A.8}$$

Because ε is independent of the index i, we drop the index i in both s_i and c_i for simplicity. Then, we have s and c, whose outcomes belong to the alphabet $\mathcal{A}$. Because $\mathcal{A}$ consists of Q points, there are a total of Q^2 combinations of $\{s, c\}$, which are $\left\{s_{(n)}, c_{(m)}\right\}, n, m = 1, 2, \ldots, Q$, where $s_{(n)}$ and $c_{(m)}$ are the nth and mth elements of $\mathcal{A}$, respectively. If $n = m$, we have $s_{(n)} = c_{(m)}$; otherwise, $s_{(n)} \neq c_{(m)}$. There are Q combinations of $\left\{s_{(n)}, c_{(n)}\right\}$, which have equal probabilities of $\frac{P_0}{Q}$, where $P_0 = \mathbb{P}\{s = c\}$. Therefore, ε can be expressed as

$$\begin{aligned}
\varepsilon = \mathbb{E}\{sc\} &= \sum_{n=1}^{Q}\sum_{m=1}^{Q} s_{(n)}c_{(m)}\mathbb{P}\left\{s_{(n)}, c_{(m)}\right\} \\
&= \frac{P_0}{Q}\sum_{n=1}^{Q} s_{(n)}^2 + \sum_{n=1}^{Q}\sum_{m\neq n}^{Q} s_{(n)}c_{(m)}\mathbb{P}\left\{s_{(n)}, c_{(m)}\right\} \\
&= \sigma_t^2 P_0 + \sum_{n=1}^{Q}\sum_{m\neq n}^{Q} s_{(n)}c_{(m)}\mathbb{P}\left\{s_{(n)}, c_{(m)}\right\} .
\end{aligned} \tag{A.9}$$

In the TS algorithm, the searching process starts from an initial solution that is usually a linear solution such as ZF or MMSE, and it is close to the true solution at high SNRs. Furthermore, as the iteration proceeds, it is likely that the TS solution gets closer to $\boldsymbol{s}$. Therefore, at high SNRs, we have $P_0 \gg \mathbb{P}\left\{s_{(n)}, c_{(m)}\right\}$, with $n \neq m$, i.e., $P_0 \approx 1, \mathbb{P}\left\{s_{(n)}, c_{(m)}\right\} \approx 0$. We note that large MIMO typically works in the high-SNR region. Therefore, we take the approximation of $\mathbb{P}\left\{s_{(n)}, c_{(m)}\right\} \approx 0, n \neq m$, and from (A.9), we have $\varepsilon \approx \sigma_t^2$.

Therefore, from (A.8), $\mathbb{E}\left\{|z_n|^2\right\}$ can be approximated as

$$\mathbb{E}\left\{|z_n|^2\right\} \approx \sigma_v^2 \sum_{i=1}^{N} |q_{n,i}|^2 .$$

Because $\boldsymbol{Q}^T$ is a unitary matrix, its rows have the unit squared-norm, i.e., $\|\boldsymbol{q}_n\|^2 = \sum_{i=1}^{N} |q_{n,i}|^2 = 1$. Consequently, we obtain

$$\mathbb{E}\left\{|z_n|^2\right\} = \sigma_v^2. \tag{A.10}$$

Now, we consider the second expectation in (A.2). From (A.3), we get

$$\begin{aligned}
\mathbb{E}\left\{z_n \delta_d\right\} &= \mathbb{E}\left\{\left[\boldsymbol{p}_n^T (\boldsymbol{s} - \boldsymbol{c}) + \boldsymbol{q}_n^T \boldsymbol{v}\right] \delta_d\right\} \\
&= \mathbb{E}\left\{\boldsymbol{p}_n^T \boldsymbol{s} \delta_d\right\} - \mathbb{E}\left\{\boldsymbol{p}_n^T \boldsymbol{c} \delta_d\right\} + \mathbb{E}\left\{\boldsymbol{q}_n^T \boldsymbol{v} \delta_d\right\} \\
&= \sum_{i=1}^{N} r_{n,i} \left(\mathbb{E}\left\{s_i \delta_d\right\} - \mathbb{E}\left\{c_i \delta_d\right\}\right) + q_{n,i} \mathbb{E}\left\{v_i \delta_d\right\} \\
&= \sum_{i=1}^{N} r_{n,i} \left(\mathbb{E}\left\{s_i \left(c_d - x_d\right)\right\} - \mathbb{E}\left\{c_i \left(c_d - x_d\right)\right\}\right) + q_{n,i} \mathbb{E}\left\{v_i \delta_d\right\} \\
&= \sum_{i=1}^{N} r_{n,i} \left(\mathbb{E}\left\{\left(s_i - c_i\right)\left(c_d - x_d\right)\right\}\right) + q_{n,i} \mathbb{E}\left\{v_i \delta_d\right\} .
\end{aligned} \tag{A.11}$$

In (A.11), $\mathbb{E}\left\{v_i \delta_d\right\} = 0$ because $\mathbb{E}\left\{v_i\right\} = 0$ and v_i is independent of δ_d. We note that s_i, c_i, and x_d are discrete random variables with zero means, i.e., $\mathbb{E}\left\{s_i\right\} = \mathbb{E}\left\{c_i\right\} = \mathbb{E}\left\{x_d\right\} = 0$. For $i \neq d$, s_i and c_i are independent of x_d and c_d, which leads to $\mathbb{E}\left\{s_i c_d\right\} = \mathbb{E}\left\{s_i x_d\right\} = \mathbb{E}\left\{c_i c_d\right\} = \mathbb{E}\left\{c_i x_d\right\} = 0$. Consequently, (A.11) is reduced to

$$\mathbb{E}\left\{z_n \delta_d\right\} = r_{n,d} \mathbb{E}\left\{\left(s_d - c_d\right)\left(c_d - x_d\right)\right\} . \tag{A.12}$$

Because $\mathbb{P}\{s_d = c_d\} = P_0 \approx 1$ at high SNRs, from (A.12), we can write

$$\mathbb{E}\{z_n \delta_d\} \approx 0. \tag{A.13}$$

From (A.2), (A.10), and (A.13), we obtain (2.11) in Lemma 1.

B. Proof of Lemma 2.2

The average metric of a neighbor can be expressed as

$$\mathbb{E}\left\{\phi\left(\boldsymbol{x}\right)\right\} = \sum_{n=1}^{d} \mathbb{E}\left\{\left|z_n + r_{n,d}\delta_d\right|^2\right\} + \sum_{n=d+1}^{N} \mathbb{E}\left\{\left|z_n\right|^2\right\}. \tag{B.1}$$

By inserting (A.10) and (2.11) into (B.1), we obtain

$$\begin{aligned}
\mathbb{E}\left\{\phi\left(\boldsymbol{x}\right)\right\} &= \sum_{n=1}^{d} \left(\sigma_v^2 + \left|r_{n,d}\right|^2 \delta^2\right) + \sum_{n=d+1}^{N} \sigma_v^2 \\
&= N\sigma_v^2 + \delta^2 \sum_{n=1}^{d} \left|r_{n,d}\right|^2 \\
&= N\sigma_v^2 + \delta^2 \left\|\boldsymbol{r}_d\right\|^2.
\end{aligned}$$

By exploiting the property $\|\boldsymbol{h}_d\| = \|\boldsymbol{r}_d\|$, we obtain (2.12) in Lemma 2.

References

[1] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, "Energy and spectral efficiency of very large multiuser MIMO systems," *IEEE Trans. Commun.*, vol. 61, no. 4, pp. 1436–1449, 2013.

[2] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Trans. Wireless Commun.*, vol. 9, no. 11, pp. 3590–3600, 2010.

[3] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "Large-scale MIMO detection for 3GPP LTE: Algorithms and FPGA implementations," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 916–929, 2014.

[4] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1474–1484, 2005.

[5] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 491–503, 2006.

[6] J. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, 1984.

[7] L. G. Barbero and J. S. Thompson, "Fixing the complexity of the sphere decoder for MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, 2008.

[8] S. Han and C. Tellambura, "A complexity-efficient sphere decoder for MIMO systems," in *IEEE International Conf. on Commun. (ICC)*, 2011, pp. 1–5.

[9] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, 2013.

[10] N. Srinidhi, T. Datta, A. Chockalingam, and B. S. Rajan, "Layered tabu search algorithm for large-MIMO detection and a lower bound on ML performance," *IEEE Trans. Commun.*, vol. 59, no. 11, pp. 2955–2963, 2011.

[11] A. Chockalingam and B. S. Rajan, *Large MIMO systems.* Cambridge University Press, 2014.

[12] M. Mandloi and V. Bhatia, "Low-Complexity Near-Optimal Iterative Sequential Detection for Uplink Massive MIMO Systems," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 568–571, 2017.

[13] K. V. Vardhan, S. K. Mohammed, A. Chockalingam, and B. S. Rajan, "A low-complexity detector for large MIMO systems and multicarrier CDMA systems," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 3, pp. 473–485, 2008.

[14] S. K. Mohammed, A. Zaki, A. Chockalingam, and B. S. Rajan, "High-rate space–time coded large-MIMO systems: Low-complexity detection and channel estimation," *IEEE J. Sel. Topics Signal Proces.*, vol. 3, no. 6, pp. 958–974, 2009.

[15] X. Qin, Z. Yan, and G. He, "A near-optimal detection scheme based on joint steepest descent and Jacobi method for uplink massive MIMO systems," *IEEE Commun. Lett.*, vol. 20, no. 2, pp. 276–279, 2016.

[16] M. Mandloi and V. Bhatia, "Error Recovery Based Low-Complexity Detection for Uplink Massive MIMO systems," *IEEE Wireless Commun. Lett.*, 2017.

[17] P. Som, T. Datta, A. Chockalingam, and B. S. Rajan, "Improved large-MIMO detection based on damped belief propagation," in *IEEE Workshop on Inf. Theory (ITW)*, 2010, pp. 1–5.

[18] S. K. Mohammed, A. Chockalingam, and B. S. Rajan, "Low-complexity near-map decoding of large non-orthogonal stbcs using pda," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on.* IEEE, 2009, pp. 1998–2002.

[19] T. Datta, N. A. Kumar, A. Chockalingam, and B. S. Rajan, "A novel Monte-Carlo-sampling-based receiver for large-scale uplink multiuser MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 62, no. 7, pp. 3019–3038, 2013.

[20] M. Hansen, B. Hassibi, A. G. Dimakis, and W. Xu, "Near-optimal detection in MIMO systems using Gibbs sampling," in *IEEE Global Telecommun. Conf. (GLOBECOM)*, 2009, pp. 1–6.

[21] M. Mandloi and V. Bhatia, "Layered Gibbs Sampling Algorithm for Near-Optimal Detection in Large-MIMO Systems," in *IEEE Wireless Commun. and Networking Conf. (WCNC)*, 2017, pp. 1–6.

[22] T. L. Narasimhan and A. Chockalingam, "Channel hardening-exploiting message passing (CHEMP) receiver in large-scale MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 847–860, 2014.

[23] P. Švač, F. Meyer, E. Riegler, and F. Hlawatsch, "Soft-heuristic detectors for large MIMO systems," *IEEE Trans. Signal Process.*, vol. 61, no. 18, pp. 4573–4586, 2013.

[24] N. Srinidhi, S. K. Mohammed, A. Chockalingam, and B. S. Rajan, "Low-complexity near-ML decoding of large non-orthogonal STBCs using reactive tabu search," in *IEEE International Symposium on Inf. Theory.*, 2009, pp. 1993–1997.

[25] ——, "Near-ML signal detection in large-dimension linear vector channels using reactive tabu search," *arXiv preprint arXiv:0911.4640*, 2009.

[26] T. Datta, N. Srinidhi, A. Chockalingam, and B. S. Rajan, "Random-restart reactive tabu search algorithm for detection in large-MIMO systems," *IEEE Commun. Lett.*, vol. 14, no. 12, pp. 1107–1109, 2010.

[27] H. Zhao, H. Long, and W. Wang, "Tabu search detection for MIMO systems," in *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Commun.*, 2007, pp. 1–5.

[28] K. S. Gyamfi, J. Baek, and K. Lee, "Lattice reduction aided tabu search with channel-dependent stopping criterion for MIMO systems," *Electronics Lett.*, vol. 51, no. 24, pp. 2062–2064, 2015.

[29] N. Farsad and A. Goldsmith, "Detection algorithms for communication systems using deep learning," *arXiv preprint arXiv:1705.08044*, 2017.

[30] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 114–117, 2017.

[31] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, 2019.

[32] ——, "Deep MIMO detection," *IEEE Int. Workshop Signal Process. Advances in Wireless Commun.*, pp. 1–5, 2017.

[33] G. Gao, C. Dong, and K. Niu, "Sparsely Connected Neural Network for Massive MIMO Detection," *EasyChair*, Tech. Rep., 2018.

[34] M. Mohammadkarimi, M. Mehrabi, M. Ardakani, and Y. Jing, "Deep Learning-Based Sphere Decoding," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4368–4378, 2019.

[35] N. T. Nguyen, K. Lee, and H. Dai, "QR-Decomposition-Aided Tabu Search Detection for Large MIMO Systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4857–4870, 2019.

[36] N. T. Nguyen and K. Lee, "Groupwise Neighbor Examination for Tabu Search Detection in Large MIMO systems," *to appear in IEEE Trans. Veh. Technol.*, 2019.

[37] ——, "Deep learning-aided tabu search detection for large MIMO systems," *IEEE Trans. Wireless Commun.*, 2020.

[38] A. Askri and G. R.-B. Othman, "DNN assisted Sphere Decoder," in *Int. Symp. Inf. Theory (ISIT)*. IEEE, 2019, pp. 1172–1176.

[39] D. Weon and K. Lee, "Learning-Aided Deep Path Prediction for Sphere Decoding in Large MIMO Systems," *arXiv preprint arXiv:2001.00342*, 2020.

[40] R. W. Farebrother, *Linear least squares computations*. Marcel Dekker, Inc., 1988.

[41] K. J. Kim, J. Yue, R. A. Iltis, and J. D. Gibson, "A qrd-m/kalman filter-based detection and channel estimation algorithm for mimo-ofdm systems," *IEEE Trans. Wireless Commun.*, vol. 4, no. 2, pp. 710–721, 2005.

[42] X. Dai, R. Zou, J. An, X. Li, S. Sun, and Y. Wang, "Reducing the complexity of quasi-maximum-likelihood detectors through companding for coded MIMO systems," *IEEE Trans. Veh. Tech.*, vol. 61, no. 3, pp. 1109–1123, 2012.

[43] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[44] R. Hunger, *Floating point operations in matrix-vector calculus.* Munich University of Technology, Inst. for Circuit Theory and Signal Processing Munich, 2005.

[45] R. Sedgewick, "Implementing quicksort programs," *Communications of the ACM*, vol. 21, no. 10, pp. 847–857, 1978.

[46] R. L. Graham, D. E. Knuth, and O. Patashnik, "Concrete Mathematics: A Foundation for Computer Science. 2nd," 1994.

[47] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[49] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cogn. Model.*, vol. 5, no. 3, p. 1, 1988.

[50] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proc. 19th Int. Conf. Comput. Statist.*, pp. 177–186, 2010.

[51] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Operating Systems Design and Implementation*, vol. 16, 2016, pp. 265–283.

[52] A. Paulraj, R. Nabar, and D. Gore, *Introduction to space-time wireless communications.* Cambridge university press, 2003.

[53] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Multilevel MIMO detection with deep learning," in *IEEE 52nd Asilomar Conf. Signals, Systems, and Computers*, 2018, pp. 1805–1809.

[54] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2806–2818, 2005.

[55] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *2018 IEEE Global Conf. Signal Inf. Processing (GlobalSIP)*, 2018, pp. 584–588.

[56] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, "Adaptive neural signal detection for massive MIMO," *arXiv preprint arXiv:1906.04610*, 2019.

[57] J. Ma and L. Ping, "Orthogonal amp," *IEEE Access*, vol. 5, pp. 2020–2033, 2017.

[58] A. M. Chan and I. Lee, "A new reduced-complexity sphere decoder for multiple antenna systems," in *Int. Conf. Commun.*, vol. 1. IEEE, 2002, pp. 460–464.

[59] H. Vikalo and B. Hassibi, "On the sphere-decoding algorithm ii. generalizations, second-order statistics, and applications to communications," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2819–2834, 2005.

초 록

대규모 다중입출력 시스템을 위한 근최적 신호 검출

응웬 딴 난

(지도교수 이경천)

전기정보공학과

일반대학원

서울과학기술대학교

본 논문에서는 다중사용자 대규모 다중입출력(Multiple-Input Multiple-Output, MIMO) 시스템의 중요 최적화 문제 중 하나인 저복잡도 근최적 신호 수신 방식을 다룬다. 타부 탐색(Tabu Search, TS)와 구복호(Sphere Decoding, SD)를 최적화함으로써 대규모 다중입출력 시스템을 위한 다섯 종류의 저복잡도 근최적 수신 알고리듬, 즉 QR 분해 기반 TS (QR-TS), Neighbor-Grouped TS (NG-TS) 심층학습(Deep Learning, DL) 기반 TS (DL-TS), Fast-DL 기반 SD (FDL-SD), Fast-DL 기반 K-best SD (KSD) (FDL-KSD) 방식을 제안한다.

다중입출력 시스템에 대한 종래의 타부 탐색 신호 검파 알고리듬은 최적 이웃 벡터(Neighbor Vector)를 결정하기 위해 모든 이웃 벡터의 비용 메트릭(Metric)을 계산한다. 하지만 대규모 다중입출력 시스템에서 이웃 벡터의 수와 벡터 차원이 매우 크기 때문에 이 방식은 지나치게 높은 계산 복잡도를 요구하게 된다. 제안하는 QR-TS 알고리듬은 해일 확률이 낮은 이웃 벡터를 조기에 제거함으로써 모든 이웃의 메트릭을 계산하지 않고도 최적의 이웃을 찾을 수 있도록 한다. 한편, NG-TS 알고리듬은 단순화된 비용 함수를 사용하여 이웃 벡터들을 그룹으로 나누어 최적 이웃 벡터를 찾는다. QR-TS 및 NG-TS 알고리듬의 계산복잡도를 추가 감소시키기 위하여 최적 순서화 방식도 제안한다. 모의실험 결과에서 QR-TS는 기존 타부

탐색 대비 약 $60-90\%$의 복잡도 감소를 달성하며, NG-TS는 기존 타부 탐색 방식 대비 성능 손실 없이 최대 85%의 복잡도 감소를 달성하는 것으로 나타났다.

제안된 DL 기반 신호 수신 방식, 즉 DL-TS, FDL-SD 및 FDL-KSD은 고속수렴 희소접속 검파 네트워크(Fast-Convergence Sparsely Connected Detection Network, FS-Net)를 각각 TS, SD 및 KSD 방식과 결합한다. DL-TS 알고리듬에서는 FS-Net에 의해 얻어진 해를 초기해로 이용하며, 최적해에 조기에 도달할 수 있도록 적응적 조기 종료 알고리듬 및 수정된 검색 방식을 이용한다. FDL-SD 및 FDL-KSD 알고리듬에서는 FS-Net을 이용해 높은 정확도의 초기해를 생성하고 후보해 및 계층 순서화, 조기 제거 방식을 이용함으로써 SD 및 KSD의 검색 속도를 향상시킨다. 모의실험 결과를 통해 제안 알고리듬이 기존 방식 대비 성능 손실이 없거나 매우 적은 손실만 가지면서도 90% 이상의 계산 복잡도 감소 이득을 얻는 것을 확인하였다.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my Advisor, Prof. Kyungchun Lee, for continuous support during these past five years. He taught me how to become an honorable researcher who should always attempt to contribute novelties to the world but should also be familiar with rejections for publications. He is my role model of a respected researcher to follow.

Besides, I am also very grateful to Prof. Huaiyu Dai for his warm welcome and advising me during my visiting research at North Carolina State University (NCSU). I would also like to thank Prof. Taehyun Jeon, Prof. Dong-ho Kim, Prof. Ji-Hoon Yun, and Prof. Illsoo Sohn for kindly participating in my thesis defense and giving insightful comments on my thesis. My sincere thanks also go to Dr. Do Trong Tuan for his kind recommendation for my studying abroad and my teachers, Ms. Phuong, Luong, Giang, Trinh, and Huong, who devotedly taught me from early stages of my academic career.

I am thankful to all of my SeoulTech friends and lab mates. I especially thank Le Anh, Hung, Hinh, Luu, Thuy, Nam, Tung, Dung, Viet, Quynh Mai, and Minh Giang for being next to me during my good and bad times. I would also like to thank Cuong and Minh for making my visiting research at NCSU a memorable trip. In particular, I would like to thank Huy for his enthusiasm and kindly sharing everything with me.

Last but not the least, I must express my very profound gratitude to my parents, brother, sister, nephews, and niece for providing me with unfailing support and spiritual encouragement throughout my years of study. This accomplishment would not have been possible without them.