循环码的神经网络信念传播解码算法研究

重庆大学硕士学位论文

(学术学位)

作者姓名：王　铭

指导教师：黎　勇 教授

学科门类：工　学

学科名称：计算机科学与技术

研究方向：机器学习

答辩委员会主席：李传东

授位时间：2023 年 6 月

Study on the Neural Network Belief-Propagation Decoding Algorithms for Cyclic Codes

A thesis submitted to Chongqing University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Computer Science and Technology

by

Ming Wang

Supervisor: Prof. Yong Li

June, 2023

摘 要

二进制循环码例如平方剩余 (quadratic residue, QR) 码和 Bose-Chaudhuri-Hocquenghem (BCH) 码等在物理层通信中有着重要的应用价值。与现在广泛使用的低密度奇偶校验 (low-density parity-check, LDPC) 码相比，BCH 码和 QR 码均为高密度奇偶校验 (high-density parity-check, HDPC) 码。此外，QR 码拥有严格的数学结构以及经过证明的最小汉明距离下界，而 BCH 码和 Reed-Muller (RM) 码等有着确定的最小汉明距离。因此，这些循环码在短码场景下通常拥有更好的理论性能。然而，目前存在的解码器难以在保持低复杂度的同时达到此类码字的理论性能。为解决该问题，本文以改善解码性能和降低解码复杂度为目标，提出了基于信念传播 (belief-propagation, BP) 算法的神经网络解码器，论文主要的创新点如下：

（1）提出了使用神经网络 BP 算法配合 modified random redundant decoding (mRRD) 算法对 QR 码进行解码，并提出根据小型陷阱集挑选循环移位，从而在解码准确率仅略微下降的情况下降低了 mRRD 算法的复杂度。此外，还提出了基于 Chase-II 算法的神经网络解码器，进一步提升了解码准确率，在 QR (47, 24) 码上获得了接近 ML 解码的帧错误率性能。

（2）提出了基于分层最小和 (min-sum) 算法的神经网络解码器，在解码准确率有所改进的同时降低了解码迭代的复杂度。提出的分层神经网络 min-sum 算法通过将权重分配规则从逐边分配变为逐迭代分配，大大削减了网络规模，从而使得提出的解码器更加易于训练且占用的硬件资源更低，方便在电路上实现。此外，将分层神经网络 min-sum 算法与 mRRD 算法相结合可进一步提高解码器的解码准确率；同时引入了快速终止条件，在解码性能逼近 ML 解码性能的情况下大大减少了 mRRD 算法的平均迭代次数。

实验表明：本文所提出的用于循环码的神经网络解码算法在短码场景下可以获得接近 ML 解码的性能。

关键词：循环码；深度神经网络；信念传播算法；迭代解码；最大似然解码

Abstract

Binary cyclic codes such as quadratic residue (QR) codes and Bose-Chaudhuri-Hocquenghem (BCH) codes are important in physical layer communications. Compared with the widely used low-density parity-check (LDPC) codes, both BCH codes and QR codes are mostly high-density parity-check (HDPC) codes. In addition, QR codes have a strict mathematical structure and a proven minimum Hamming distance lower bound, while BCH codes and Reed-Muller (RM) codes have a determined minimum Hamming distance. Therefore, they have better theoretical performance in the scenario of short code length. However, there is a lack of high-speed decoders that can achieve the theoretical performance of these cyclic codes with a low complexity. To address this problem, this thesis proposes a neural network decoder based on the belief-propagation (BP) algorithm to improve decoding performance and reduce decoding complexity, and the main contributions of the thesis are as follows.

(1) We propose to use neural network decoders based on the belief-propagation algorithm with the modified random redundant decoding (mRRD) algorithm to decode QR codes. We modify the original mRRD by selecting the optimal cyclic shift based on small trapping sets to reduce the implementation difficulty and complexity without significantly reducing the decoding accuracy. Besides, a neural network decoder based on the Chase-II algorithm is proposed to further reduce the error rate. Simulation results show that it can approach ML performance for the QR (47, 24) code.

(2) A neural network decoder based on the layered min-sum algorithm is proposed to reduce decoding iterations while improving the decoding accuracy. The proposed layered min-sum algorithm reduces the network size by changing the weight assignment rule from edge-wise to iteration-wise, making the proposed decoder easier to train and consuming fewer hardware resources for implementation on circuits. In addition, the mRRD algorithm is combined to further improve the decoding accuracy of the decoder, and the early stopping criterion is introduced to greatly reduce the average number of iterations of the mRRD algorithm under the condition that decoding accuracy approaches ML performance.

Experiments conducted on the QR (47, 24) code, BCH (63, 45) code, and punctured RM (127, 99) code show that the proposed method can approach the ML decoding performance in short code length scenarios.

Key words: cyclic codes; deep neural networks; belief-propagation; iterative decoding; maximum-likelihood decoding

目 录

1 绪 论

1.1 研究背景及意义

1.1.1 循环码的神经网络解码器的选题背景

超可靠低时延通信 (ultra-reliable low latency communications, URLLC) 是第五代通信系统 (5th generation, 5G) 中的三大场景之一，其目的是为了向自动驾驶、物联网、远程工业控制等场景提供基础设施。截止到目前，URLLC 的技术路线仍在讨论之中。第三代合作伙伴计划 (3rd generation partnership project, 3GPP) 对于 URLLC 的期望是在 99.999%的可靠性下实现小于 1 毫秒的通信延迟[1]。一般来讲，可靠性和时延是一对相互矛盾的指标，因为增加可靠性常常需要增加编码冗余或是进行信息重传，这些都会带来额外的延迟。由于 URLLC 场景对于通信延迟和可靠度都有着极高的要求，这就对通信所使用的信道编码提出了很大的挑战。信道编码也叫做前向纠错码，其原理是信息的发送方会在发送的信息中添加一些冗余，而接收方通过这些冗余信息判断在传输过程中是否发生错误并对其进行纠正。目前的 5G 中的增强型移动宽带 (enhanced mobile broadband, eMBB) 场景分别使用了低密度奇偶校验码 (low-density parity-check, LDPC) 码[2]以及 Polar 码[3]作为数据信道和控制信道的信道编码方案。想要达到 URLLC 的技术指标，编码要有较好的纠错性能且码长不能太长，因为长码会带来更高的时延。尽管 LDPC 码性能优异且技术非常成熟，但其缺点在于码长较短时性能表现一般。而长度较短的循环码（比如 BCH 码）在最大似然 (maximum-likelihood, ML) 解码下拥有优异的性能，这使其成为了 URLLC 场景下信道编码的有力竞争者。然而包括循环码在内的大多数线性分组码的 ML 解码是一个非确定性多项式 (non-deterministic polynomial-time, NP) 问题，所以想要精确达到这些码的 ML 性能几乎不可能。因此，研究者希望能在复杂度显著降低的情况下逼近 ML 性能。

随着计算机性能的不断提高以及神经网络被重新发现，目前深度神经网络已经成为了很多棘手问题的最佳解决方案，很多过去难以解决的问题已经被深度神经网络解决或取得了重大进展。例如可以打败人类顶尖棋手的围棋 AI（AlphaGo[4]等），高质量的对话文本生成（ChatGPT[5]、LaMDA[6]等）和各种风格的图像生成（Diffusion model[7]等）。纵观这些问题，主要可以分为两类，第一类是拥有海量状态空间和计算复杂度的问题，例如围棋 AI 等；第二类则是没有精确定义的问题，例如文本生成和图像生成等。而循环码的 ML 解码问题就可以归类为第一类问题。尽管 ML 解码是一个有着明确数学定义的问题，但如果使用暴力搜索的方法对信息位长度为k的循环码进行解码，在最差情况下需要搜索2^k次才能达到 ML 解码的

性能。因此我们很自然的想到使用神经网络来减少复杂度并获得近似于 ML 解码的性能从而达到 URLLC 对可靠性的要求。同时我们还希望降低神经网络解码器的规模从而达到 URLLC 对于延迟的要求。除了延迟方面的问题，复杂的神经网络常常会要求更高的功耗，难以应用到移动设备等计算能力受限或对功耗控制十分严格的设备中。

1.1.2 循环码的神经网络解码器的研究意义

循环码作为 URLLC 中编码技术的一个有力候选者，在学术界和工业界均拥有着很高的研究价值。URLLC 中的低延迟特性要求编码的长度不能过长，而成熟的 LDPC 等编码方案在该长度性能表现一般。BCH 码和 QR 码等循环码在中短码长下有着优秀的性能，但这些循环码缺乏能够高效逼近 ML 性能的解码器。神经网络循环码解码器若能克服该问题，则可以在通信过程中带来以下好处：

（1）更低的解码延迟：降低编解码延迟是实现低延迟通信的重中之重。循环码可以使用较短的码长维持满意的性能，从而降低延迟；而一个好的解码器可以用较少的运算解码接收数据并降低延迟。更低的通信延迟则可以支撑更多样的应用场景，例如工业实时控制、远程游戏等。目前的循环码解码算法都比较复杂，而神经网络有望解决这一点。

（2）更低的误码率：好的解码器同样能提供更低的误码率，从而降低通信开销，提高通信的可靠性。这将使得对可靠性要求很高的应用，例如远程医疗、无人机控制等等可以通过远程无线网络进行。目前的循环码解码算法几乎都为误码率较高的硬判决算法，而加入软判决后的复杂度又难以控制。神经网络循环码解码器则可以在利用软信息降低误码率的同时降低复杂度。

（3）更低的功耗：现代手机的整体功耗其中有相当一部分是通信基带所带来的，而更好的解码器则可以降低功耗，提高手机续航能力的同时保护环境。

综上，复杂度和误码率更低的循环码解码器在多个领域中都有着广阔的应用前景以及很大的经济价值，而神经网络循环码解码器则有可能做到兼顾复杂度和误码率。

1.2 研究历史及现状

循环码是一类非常特殊的线性分组码，其特点在于任意一个码字在进行循环移位后依然是一个码字，这使得相应的编码可以使用循环移位寄存器高效实现。循环码通常拥有丰富的数学结构且用途十分广泛。很多著名的编码均为循环码，例如 BCH 码、QR 码、punctured RM 码、Hamming 码等。然而，大部分循环码都缺少能够高效逼近其 ML 性能的解码算法。例如二进制 QR 码由 Prange 在 1958 年提出[8]，其码长约为 1/2 且在同码长和码率下通常拥有较大的最小汉明距离。一般

而言对于任意一个线性分组码，其最小汉明距离越大，理论性能就越好。事实上码长为n的 QR 码的最小汉明距离至少是$\sqrt{n}$，这被称为平方根界。这使得 QR 码是目前已知性能较好的二进制编码之一。尽管 QR 码理论上拥有优秀的性能，但其解码算法一直是学界公认的难题。Berlekamp 在他的著作中也曾认为 QR 码是一类难以解码的好码[9]361。此外，QR 码具有较大的置换群，Gleason–Prange 定理[10]证明了扩展 QR 码至少有一个与射影特殊线性群$PSL_2(n)$或特殊线性群$SL_2(n)$群同构的子群，这使得一些 QR 码的解码算法会从置换入手[11]。由于优秀解码算法的匮乏，目前投入应用的 QR 码码长都较短。例如 QR (7, 4) 码被广泛应用于计算机的纠错内存上 (error-correcting code memory, ECC memory)，(24, 12) 扩展 QR 码（也称为扩展格雷码）已经被应用在多种通信链路中，例如“旅行者”的影像系统[12]以及高频广播系统[13]。Li 等人于 2018 年提出了基于查表法的 DS 算法[14]。该算法可适用于所有码长的 QR 码，且在码长较短时速度极快。但是即便是目前较快的 DS 算法，其复杂度随着码长的增长也会以极高的速度增长。对于码长超过 100 的 QR 码，DS 算法几乎没有实用性。BCH 码和 RM 码的问题则和 QR 码略有不同。BCH 码[15][16]由 Hocquenghem、Bose、和 Ray-Chaudhuri 三人提出和完善，并以三人名字的首字母命名。目前 BCH 码最常用的解码算法是 BM 算法，在 1965 年由 Berlekamp 提出[9]176-195 并在 1969 年被 Massey 改进[17]。RM 码由 Muller 在 1954 年发明[18]，其解码算法于同年由 Reed 提出[19]。BCH 码和 RM 码相比 QR 码而言是更易解的编码，其解码算法速度很快，并且目前投入使用的编码长度相比 QR 码更长。但是和 DS 算法一样，BCH 码和 RM 码的常用解码算法均为硬判决算法，因此其性能距离 ML 解码性能较远。若使用 Chase 算法等软判决算法对其进行增强，则可以在部分短码上达到接近 ML 解码的性能，但是其复杂度相比原硬判决算法会成倍增长。此外，即使使用了 Chase 算法，在码长稍微长一些的时候其性能离 ML 解码性能仍有较大差距。而机器学习与深度学习领域的进步则为该问题带来了新的解决思路。

近年来学术界提出了多种使用深度学习设计或解码信道编码的方法。尽管针对循环码设计的神经网络解码器并不多，但是由于循环码同时也属于线性分组码，其中的经验也值得我们借鉴。Kim 等人[20]1-17 证明了循环神经网络 (recurrent neural network, RNN) 可以用来对卷积码和 Turbo 码进行解码。Zhang 等人[21]426-230 和 Jiang 等人[22]-[23]的论文中使用了自编码器同时学习编码和解码算法。Huang 等人[24]使用强化学习和遗传算法设计二进制线性分组码，其设计的码字错误率性能达到甚至超过了传统的编码方案。但是由于遗传算法所设计出的码字缺乏数学结构，导致其只能使用复杂度较高的通用解码算法进行解码。Gruber[25]等人观察到神经网络对于结构化的码字拥有较强的泛化能力，对于训练集中没有出现过的码字也可以

完成解码。然而，该文献所提出的神经网络解码器性能较差，这可能是因为其使用的多层感知机 (multi-layer perceptron, MLP) 神经网络只学习到了编码的部分结构。纵观上文所提及以及未提及的工作，目前基于深度神经网络的解码器大部分都是针对线性分组码设计的，也有部分神经网络解码器针对的是卷积码或 Turbo 码，甚至是新设计的码字。一般而言，这些解码器可以分为两类：数据驱动的解码器和模型驱动的解码器。

数据驱动的神经网络解码器常常使用深度学习中的经典或热门网络模型和结构，如 RNN、自编码器等等。例如，在文献[20]1-17 中作者使用了 RNN 训练了一个可以对卷积码和 Turbo 码进行解码的解码器。此外，数据驱动的端到端自编码器也被用于同时学习编码与解码[21]-[22]。其中 Zhang 等人[21]426-430 考虑了删余的方法，引入了一种基于 RNN 的自编码器用于构建高码率的信道编码。其构造出的编码在符号间干扰 (inter-symbol interference, ISI) 信道上低信噪比区域的性能超过了 LDPC 码。在文献[22]中，作者使用基于 RNN 的自编码器学习多种码率不超过 1/2 的信道编码，且学习到的信道编码性能超过了传统的咬尾卷积码。Jiang 等人[23]也研究了基于卷积神经网络 (convolutional neural network, CNN) 的自编码器，所学到的信道编码与传统编码类似，纠错性能随码长增长而变强。尽管数据驱动的编解码研究取得了以上进展，但是纯数据驱动的设计方法主要弱点在于过高的训练复杂度以及在长码上缺乏可扩展性，因为这类方法依赖大量的随机码字作为训练样本。此外，这类方法所学习到的编码其性能优势一部分也来自连续分布的码字，这一点和传统的二进制码字也极为不同。所以这类神经网络学习到的编解码方案和现在的数字通信系统不能兼容。

模型驱动的神经网络解码器则更偏向于根据码字结构设计对应的神经网络模型。这样做的好处是网络的可解释性强，训练网络所需的样本数量小，而且训练方法易于推广到其他同类码字。例如很多针对 BCH 码的神经网络解码器对于 LDPC 码以及 QR 码同样有效，因为它们都属于线性分组码。有相当一部分模型驱动的神经网络解码器是在 BP 类解码算法（包括 sum-product 算法和 min-sum 算法等）的基础上设计的，因为 BP 类解码算法通过在 Tanner 图上反复传递消息从而完成解码，而且 BP 类算法是一种迭代解码算法，这一过程易于推广到神经网络中。Nachmani 等人[26]最先将 sum-product 算法与神经网络结合并提出了神经网络 BP 解码器用于对 BCH 码进行解码。该方法通过对 Tanner 图的每条边赋予不同的权重再对这些权重进行训练，在 BCH 码上获得了远低于原版 BP 算法的解码错误率。神经网络 sum-product 算法在训练过程中可以使用全零码字和随机生成的噪声作为训练集，这是因为神经网络的连接方式和激活函数与 sum-product 算法的步骤一一对应，从原理上避免了过拟合的发生。在此基础上，Lugosch 等人[27]使用了 BP 算法

的一个变种，即 min-sum 算法，代替原来的 sum-product 算法从而形成了神经网络 min-sum 算法。该算法大幅度简化了每一次迭代所需的运算量并基本保持了神经网络 sum-product 算法的性能。而文献[28]中作者进一步研究了量化位数对神经网络 min-sum 解码算法的影响，并提出了一种非均匀量化的方法。该方法使用较少的量化位数同时保持了较好的性能。Nachmani 等人[29]又提出将神经网络 min-sum 中的乘法权重替换为减法权重，从而消除了神经网络解码器中的所有乘法操作，而解码准确率只有轻微的损失。在此之上，作者还提出共享每一层的权重，形成 RNN 的结构。这样做的好处是进一步削减了网络的参数量，但是在某些情况下会对解码准确率造成较大影响。该方法对 Polar 码同样也拥有较好的效果[30]。为了进一步提高解码准确率，Nachmani 等人还提出将 modified random redundant decoding (mRRD) 算法[31]和神经网络解码器相结合，将接收到的信号向量进行多次置换并对其进行解码，最终挑选出一个最好的结果。这样做的好处是随着置换次数的增加，解码错误率可以快速下降，但同时也会大幅增加解码所需的迭代次数，使得解码延迟增加。例如为了达到 BCH (63, 45) 码的 ML 解码性能，Nachmani 等人提出的神经网络 mRRD 解码器对于每个输入可能需要 12500 次迭代才能完成解码。另外 Nachmani 等人于 2019 年[32]通过使用图神经网络来学习 Tanner 图中权重的分布从而进一步改善了他们在 2018 年的结果，但提升效果有限。此外，Nachimani 等人在这项工作中使用了$\tanh^{-1}$的高阶泰勒展开代替原本的$\tanh^{-1}$函数从而避免了梯度爆炸。但是值得注意的是只有在使用 sum-product 算法的神经网络 BP 解码器中才存在$\tanh^{-1}$函数，若使用与其性能相似但更简单的神经网络 min-sum 解码器则不存在这一问题。

Chen 等人[33]1771-1780 于 2021 年提出了针对循环码的循环不变神经网络解码器，该解码器同样基于神经网络 sum-product 算法。作者利用了循环码的特性，构造了循环冗余矩阵并以此替换普通的校验矩阵。此外，作者还基于 ML 解码器在对解码循环码解码时拥有循环不变性这一特点推导了神经网络的权重共享公式，按照这一公式共享权重可以保证其解码准确率不弱于为每条边分别分配权重的方案。这样做大大减少了神经网络解码器的参数量，缩短了训练时间以及模型大小。同时作者还提出了表解码算法，进一步提升了解码准确率。在使用表解码算法对 (63, 45) BCH 码进行解码时，最大只需要 6400 次迭代便可以逼近 ML 解码准确率，相比 Nachmani 等人的 12500 次有着大幅度的降低。此后，在文献[34]中，Chen 等人针对表解码算法中表较小的情况优化了解码器的比特错误率性能，但帧错误率性能不变。然而在实际通讯中若发现帧错误则整帧都需要丢弃或重新传输，因此该方法对于 URLLC 场景的作用有限。随后，Chen 等人又在循环不变神经网络解码器的基础上提出了置换不变神经网络解码器[35]，该解码器利用扩展 BCH 码和 RM

码在仿射置换下不变的特性进一步扩展了校验矩阵的大小并设计了对应的神经网络解码器。该解码器性能好于循环不变解码器，同时复杂度也有所提高。但是该方法在大多数码字上并不能达到 ML 性能，例如文中对于 BCH (63, 45) 码将校验矩阵从原来的17×63扩充至4032×64，大大增加了复杂度但性能距离 ML 解码性能仍然较远。此外，Buchberger 等人[36]1957-1966 提出了在高度冗余的矩阵上训练神经网络 BP 解码器的方法，实验表明矩阵的冗余度越高，则越可能获得更低的误码率。其中冗余矩阵的每一行都是通过搜索对偶码的最小重量非零码字（其汉明重量等于码字的最小汉明距离）得到的。为了降低复杂度，作者还提出了修剪神经网络的方法。在神经网络训练完成后，可以修剪神经网络中贡献较小的权重并删除冗余矩阵中对应的行，从而在保证性能的情况下降低解码复杂度。该方法对于部分 RM 码得到了逼近 ML 解码的性能，但是需要注意的是很多码的最小汉明距离仍然是未知的。例如目前对于码长超过 200 的 QR 码，其最小汉明距离学界只有粗略的估计而无精确值。并且即使是对于已知最小汉明距离的码，想要穷举所有码重等于最小汉明距离的码字也是极其复杂的过程，例如对于 RM (256, 93) 码，其最小汉明距离为 32，拥有共 777 240 个汉明重量等于 32 的码字。想要搜索这些码字并非易事。另外，随着矩阵冗余度的增加，解码复杂度也随之增加，以上缺点使得该方法只适用于码长极短的码。

1.3 主要工作与创新

本文主要针对提升二进制循环码的解码准确率和降低神经网络解码器的运算次数展开研究。现有的部分神经网络解码器在一些编码上已经可以获得不错的解码准确率，但只能在极少部分码字上获得 ML 译码性能，因此仍有改进空间。此外，目前的神经网络解码器为了获得较好的解码准确率需要大量的训练参数以及迭代次数，这使得其难以应用到移动设备中。故本文的主要工作为在进一步提高神经网络解码器解码准确率的同时降低其复杂度。另外，为了衡量解码器的准确率，我们需与 ML 解码器的准确率进行对比。BCH 码的 ML 解码准确率已经被大量研究且有成熟的数据库，但针对 QR 码和 punctured RM 码的研究较少。故本文提出了一种基于线性规划和整数规划的 ML 解码器以便快速获得 QR 码和 punctured RM 码的 ML 解码错误率曲线。本文的主要创新点如下：

（1）提出使用神经网络 min-sum 算法对 QR 码进行解码，并基于陷阱集挑选循环移位以及结合 mRRD 提升其解码准确率。提出的 NCmRRD 算法应用于多个码字时，解码准确率大大超过 DS 算法。此外我们进一步结合了 Chase-II 算法，提出了 Chase-II-mRRD 算法，逼近了 QR (47, 24) 码的 ML 解码性能。

（2）提出了一种基于分层 min-sum 算法的神经网络解码器，所提出的 NLMS

解码器大大减少了参数量以及训练难度。在多种码字上进行的测试表明提出的解码器有着优异的解码准确率以及更低的计算复杂度。

（3）将提出的分层 min-sum 解码器与 mRRD 算法相结合，并引入了快速终止条件。新提出的 FNmRRD 解码器成功逼近了 QR (47, 24) 码、BCH (63, 45) 和 punctured RM (127, 99) 码的 ML 解码准确率，且解码所需的最大迭代次数和平均迭代次数相较其他解码器大大减少。

（4）提出了一种通过仿真获得线性分组码的 ML 解码帧错误率的算法。该算法基于线性规划和整数规划，适用于中短码且速度较快。

1.4 本文结构

本文主要涉及循环码的神经网络解码算法优化，论文的具体结构如下：

第 1 章，首先介绍本课题的选题背景与研究意义，随后简述循环码和神经网络解码器的研究历史和现状，在此基础上引出本文的研究内容和创新之处。最后介绍全文的结构安排。

第 2 章，介绍通信以及各种循环码的相关理论背景知识，包括通信使用的模型，QR 码、BCH 码、punctured RM 码的构造等等；然后介绍了 BP 类算法和陷阱集的相关知识，包括 sum-product 和 min-sum 算法的迭代公式及其各类变种、陷阱集的定义及其搜索算法。最后介绍了神经网络的相关知识。

第 3 章，分析了多种可能会影响 BP 类算法用于循环码解码时的误码率性能的机制，介绍了基于 BP 类算法的神经网络解码器，包括对数域的 sum-product 和 min-sum 算法。介绍了基于 mRRD 的神经网络解码算法，包括 mRRD 的原理并提出使用循环移位对其进行简化。最后，分别针对多种码字训练了对应的神经网络解码器，并进行了仿真实验和复杂度分析。

第 4 章，介绍了 ML 解码器和线性规划解码器的基本原理，并引出利用线性规划和整数规划加速 ML 解码的算法。接着引入快速终止条件以及 Chase-II 算法进一步提升神经网络循环 mRRD 算法的性能。提出了基于分层 min-sum 算法的神经网络解码器，将前文中的 mRRD 的神经网络解码算法、快速终止条件与所提出的神经网络分层 min-sum 算法结合。接着通过大量实验，将所提出的算法和目前最先进的神经网络解码算法做了对比，分析了其解码准确率以及计算复杂度。

第 5 章为总结与展望。总结和分析本文中基于神经网络解码器所做的研究工作，并在此基础上展望了未来可能的研究方向。

2 编码以及神经网络相关理论

本章主要介绍基本的通信模型以及编码相关的理论，以便后续章节的叙述。具体内容有信道以及噪声模型、有限域以及有限域上的基本算术操作、代数码的基本概念。此外还介绍了 QR 码、BCH 码、和 punctured RM 码的编码方法以及 BP 类算法的原理等。

2.1 通信模型

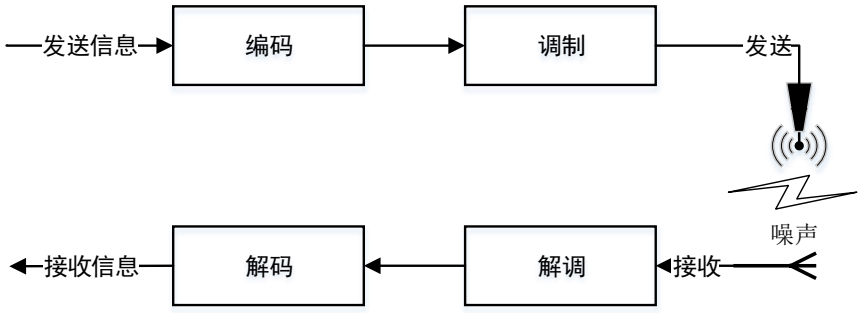

图 2.1 通信系统模型

Fig. 2.1 The model of communication systems.

本文中使用的通信模型为二进制输入加性高斯白噪声信道 (binary-input additive white gaussian noise channel, BI-AWGNC)，通信模型为图 2.1 所示。在该模型中，发送端向发送信息添加冗余，接收端收到的信息会被噪声污染，而解码就是利用冗余信息从带有噪声的接收数据中恢复发送端消息的过程。其中发送端发送的k位二进制信息记为$\mathbf{s}=(s_1,s_2,\dots,s_k)\in\{0,1\}^k$,加入冗余后得到的$n$位符号记为$\mathbf{u}=(u_1,u_2,\dots,u_n)\in\{0,1\}^n$。BI-AWGNC 中调制方式使用二进制相移键控 (binary phase-shift keying, BPSK) 调制。假设调制后的每符号能量E_s是 1，即将0调制为1，1调制为-1，则调制后的信号可以表示为$\mathbf{y}=1-2\mathbf{u}$。BI-AWGNC 中产生的噪声为高斯白噪声，记为$\mathbf{n}$，其中每个符号上收到的噪声$n_i\sim\mathcal{N}(0,\sigma^2)$，$\mathcal{N}(0,\sigma^2)$表示均值为$0$，标准差为$\sigma$的高斯分布。解码器收到的输入信号$\mathbf{r}=\mathbf{y}+\mathbf{n}$。设$E_b$为每比特能量，$N_0$为噪声的功率谱密度，信噪比$E_b/N_0$ (dB) 和噪声标准差σ之间的关系如式(2.1)所示,

$$\begin{aligned} E_s\times n &= E_b\times k,\\ N_0 &= 2\sigma^2,\\ \frac{E_b}{N_0}(\mathrm{dB}) &= 10\log_{10}\frac{E_b}{N_0}. \end{aligned} \tag{2.1}$$

即：

$$\sigma = \sqrt{\frac{1}{2\frac{k}{n}10^{\frac{\frac{E_b}{N_0}(\mathrm{dB})}{10}}}} \tag{2.2}$$

在这个过程中发送端发送的$+1$（调制前为 0）可能会由于噪声污染变为一个小于0的实数（解调后为1），而发送的-1（调制前为 1）也有可能在过程中发生类似的翻转。如图 2.2 所示，深灰色区域左侧为 1 由于噪声污染翻转为 0 的概率，右侧则为 0 翻转为 1 的概率。假设发送端发送 1 和 0 的概率相同，则在不使用任何编码时的比特错误率为：

$$\begin{aligned} P_{err} &= \Pr(s=0)\Pr(r<0|s=0) \\ &\quad + \Pr(s=1)\Pr(r>0|s=1) \\ &= \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-1)^2}{2\sigma^2}} dx \end{aligned} \tag{2.3}$$

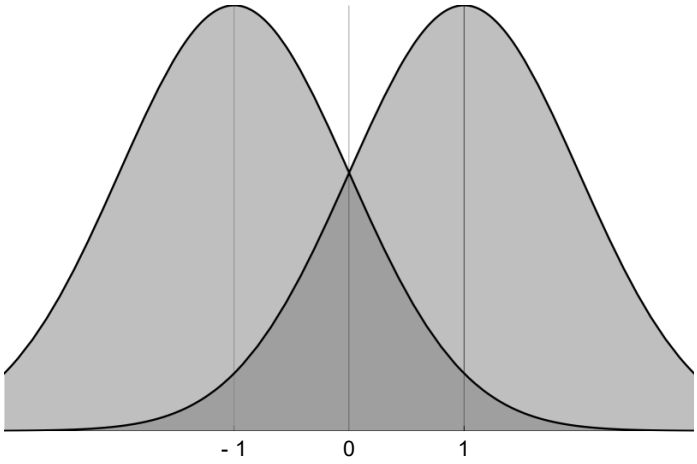

图 2.2 BI-AWGN 信道接收信号的概率分布

Fig. 2.2 Probability distribution for the received signal in BI-AWGNC.

2.2 循环码相关理论

循环码的编码理论中常常需要用到大量的有限域（也称伽罗瓦域）操作。有限域$GF(2^m)$一共包含2^m个元素，其构造如下：首先向域中加入元素 0，1 以及元素α，其中α被称为有限域$GF(2^m)$上的本原元。$GF(2^m)$中的所有非零元素都可以用α的幂次表示，即$GF(2^m)=\{0,\alpha^0,\alpha^1,\dots,\alpha^{2m-2}\}$。给定任意两个元素$\alpha^i,\alpha^j$，其乘积定义为$\alpha^{(i+j) \bmod (2^m-1)}$。在此定义下的乘法和集合$\{\alpha^0,\alpha^1,\dots,\alpha^{2m-2}\}$形成了一个循环群。而有限域中的加法则需借助本原多项式构造，例如给定$GF(2^4)$的一个本原多项式为$p(x)=x^4+x+1$，那么有$p(\alpha)=0$。$GF(2^m)$中的减法与加法相同，均为异或。在计算$\alpha^2+\alpha$时借助等式$p(\alpha)=\alpha^4+\alpha+1=0$，即$\alpha^4=\alpha+1$，则有

$\alpha^2+\alpha=\alpha(\alpha+1)=\alpha\cdot\alpha^4=\alpha^5$。

除了使用符号α的幂次构造，有限域中的元素也可以使用向量和多项式构造。使用多项式构造有限域时同样需要确定一个本原多项式$p(x)$。其后，对于任意一个系数只有 0/1 的多项式$f(x)\in GF(2)[x]$，其在有限域中表示为$f(x)\bmod p(x)$。例如x^5在$GF(2^4)$中对应$x^5\bmod p(x)=x^2+x$。因此给定$GF(2^m)$中两个多项式$M(x),N(x)$，其乘积为$M(x)N(x)=M(x)N(x)\bmod p(x)$。使用多项式表示$GF(2^m)$中的元素时，元素之间的加法也是异或操作。即给定多项式$M(x)=\sum_{0\le i\le m-1}m_ix^i,N(x)=\sum_{0\le i\le m-1}n_ix^i$，多项式之和$M(x)+N(x)=\sum_{0\le i\le m-1}(m_i\oplus n_i)x^i$。向量表示法则是将多项式表示法中的所有系数提取出来作为向量。若令幂次表示法中的α对应多项式表示中的x，$GF(2^4)$中的所有元素如表 2.1 所示。

表 2.1 $GF(2^4)$中的元素

Table 2.1 Elements in $GF(2^4)$.

$GF(2^4),p(x)=x^4+x+1$		
幂次表示	多项式表示	向量表示
0	0	(0,0,0,0)
1	1	(0,0,0,1)
α	x	(0,0,1,0)
α^2	x^2	(0,1,0,0)
α^3	x^3	(1,0,0,0)
α^4	$x+1$	(0,0,1,1)
α^5	x^2+x	(0,1,1,0)
α^6	x^3+x^2	(1,1,0,0)
α^7	x^3+x+1	(1,0,1,1)
α^8	x^2+1	(0,1,0,1)
α^9	x^3+x	(1,0,1,0)
α^{10}	x^2+x+1	(0,1,1,1)
α^{11}	x^3+x^2+x	(1,1,1,0)
α^{12}	x^3+x^2+x+1	(1,1,1,1)
α^{13}	x^3+x^2+1	(1,1,0,1)
α^{14}	x^3+1	(1,0,0,1)

编码的参数往往用(n,k)表示，其中n为编码后长度（码长），k是编码前长度（信息位长度）。对于代数码如 BCH 码和 QR 码，编码后结果（码字）对应的多

项式$u(x)=s(x)g(x)$，其中$s(x)$是编码前信息向量$\mathbf{s}$对应的多项式，$g(x)$是码的生成多项式。使用不同的编码，就意味着使用了不同的生成多项式，同时也表明冗余添加方案会有所不同。例如给定 QR (7, 4) 码（也称为 (7, 4) 汉明码[37]）的生成多项式$g(x)=x^3+x+1$和$GF(2^4)$中的待发送消息(1,0,1,1)所对应的消息多项式$s(x)=x^3+x^2+1$，编码后的消息为$s(x)g(x)=x^6+x^5+x^4+x^3+x^2+x+1$，即(1,1,1,1,1,1,1)。

对于(n,k)QR 码，其码长必须满足$n=8l\pm1$，其中l是正整数。QR 码的生成多项式可以按如下方式获得。首先定义平方剩余集如(2.4)所示，其中“mod”为求余操作。

$$Q_n=\{i|i\equiv j^2 \bmod n \text{ for } 1\le j\le n-1\} \tag{2.4}$$

令m是使得$n|2^m-1$成立的最小正整数，其中“|”代表整除，α是$GF(2^m)$的本原元。那么，$GF(2^m)$中的本原n次单位根β可以表示为$\beta=\alpha^\mu$，其中$\mu=(2^m-1)/n$。QR 码的生成多项式为：

$$g(x)=\prod_{i\in Q_n}\left(x-\beta^i\right) \tag{2.5}$$

QR 码的码长一旦确定，那么其信息位长度以及最小汉明距离也随之确定。最小汉明距离是码字集合中任意两个合法码字之间的最小汉明距离（也称为码距），更大的码距往往意味着更佳的纠错性能。最小汉明距离同时也是码字集合中除全零码字外的汉明重量（即“1”的个数）最小的码字对应的汉明重量。

BCH 码的编码参数则更为灵活，在确定 BCH 码的码长后，我们可以指定一个合适的码距，从而确定一个 BCH 码。对于一个码长为n，纠错半径为t，即码距为$d=2t+1$的 BCH 码，要确保$n=2^m-1$且$m\ge3, t<2^m-1$。在构造 BCH 码的生成多项式前，首先要确定$GF(2^m)$中各个元素对应的最小多项式。令$\alpha^i\in GF(2^m)$，那么α^i对应的最小多项式$\phi_i(x)$就是使得等式$\phi_i(\alpha^i)=0$成立的次数最低的多项式。已知最小多项式，BCH 码的生成多项式如下：

$$g(x)=\mathrm{LCM}\left(\phi_1(x),\phi_2(x),\dots,\phi_{2t}(x)\right) \tag{2.6}$$

其中LCM(·)代表最小公倍式。

RM 码的码长和信息位长度由两个整数r,m确定。其码长$n=2^m$，最小汉明距离$d=2^{m-r}$，信息位长度$k=\sum_{i=0}^{r}C_m^i$。RM 码本身并非循环码，但其可以被视为一种扩展循环码。在其码字结构中存在一位比特与所有其他比特共同构成一个校验方程，因此对 RM 码进行删余操作，将该比特删除后所形成的 punctured RM 码是一种循环码。令s为一整数，$w(s)$是s的二进制展开向量中 1 的数量。Punctured RM 码的生成多项式表示如下：

$$g(x)=\prod_{\substack{1\le w(s)\le m-r-1\\ 1\le s\le 2^m-2}}\phi_s(x) \tag{2.7}$$

为了方便在计算机上批量处理，循环码也可以使用矩阵操作的形式进行编码。对任意一个(n,k)循环码，假设其生成多项式为$g(x)=a_0x^{n-k}+a_1x^{n-k-1}+\cdots+a_{n-k}$，那么其生成矩阵（大小为$k\times n$）可以写为：

$$\mathbf{G}=\begin{bmatrix} a_0 & a_1 & \cdots & a_{n-k} & 0 & \cdots & 0 & 0\\ 0 & a_0 & a_1 & \cdots & a_{n-k} & 0 & \cdots & 0\\ 0 & 0 & a_0 & a_1 & \cdots & a_{n-k} & 0 & \cdots\\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots\\ 0 & \cdots & 0 & a_0 & a_1 & \cdots & a_{n-k} & 0\\ 0 & 0 & \cdots & 0 & a_0 & a_1 & \cdots & a_{n-k} \end{bmatrix} \tag{2.8}$$

那么对于任意$1\times k$维信息向量$\mathbf{s}$，编码后的信息$\mathbf{u}$可以通过矩阵乘法$\mathbf{u}=\mathbf{sG}$得到。其中矩阵乘法的运算规则与普通的矩阵-矩阵乘法类似，但是其中的加法操作变为异或操作。接收端在对接收信息解码后得到解码结果$\hat{\mathbf{u}}$，此时可以使用校验矩阵$\mathbf{H}$验证解码结果是否为一个合法的码字。要获得矩阵$\mathbf{H}$首先要求解校验多项式$h(x)$，给定码长n和生成多项式$g(x)$，校验多项式的计算方法如下：

$$h(x)=\frac{x^n+1}{g(x)} \tag{2.9}$$

其中的除法为$GF(2)$中的多项式长除法。假设得到的$h(x)=h_kx^k+h_{k-1}x^{k-1}+\cdots+h_0$，那么校验矩阵$\mathbf{H}$为：

$$\mathbf{H}=\begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 & 0\\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0\\ 0 & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots\\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots\\ 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_0 & 0\\ 0 & 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_0 \end{bmatrix} \tag{2.10}$$

校验矩阵$\mathbf{H}$和生成矩阵$\mathbf{G}$之间满足$\mathbf{GH}^{\mathrm{T}}=0$，而编码后的码字$\mathbf{u}$和编码前的信息$\mathbf{s}$满足$\mathbf{u}=\mathbf{sG}$，所以若解码结果$\hat{\mathbf{u}}$是一个合法码字，则应该满足$\hat{\mathbf{u}}\mathbf{H}^{\mathrm{T}}=0$，利用该性质我们便可以验证解码结果的合法性。

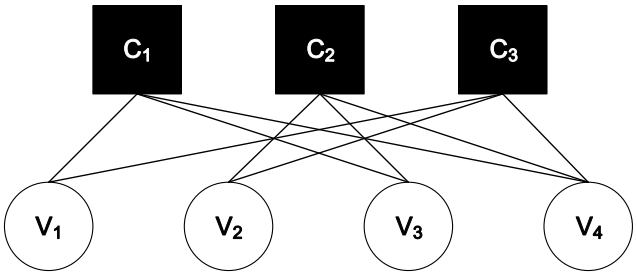

图 2.3 Tanner 图示例

Fig. 2.3 An example of Tanner graph.

校验矩阵确定了码字的校验方程组，例如给定校验矩阵$\mathbf{H}=\begin{bmatrix}1&0&1&1\\0&1&1&1\\1&1&0&1\end{bmatrix}$，等式$\hat{\mathbf{u}}\mathbf{H}^{\mathrm{T}}=0$等价于$\begin{cases}\hat{u}_1+\hat{u}_3+\hat{u}_4=0\\\hat{u}_2+\hat{u}_3+\hat{u}_4=0\\\hat{u}_1+\hat{u}_2+\hat{u}_4=0\end{cases}$，其对应的图形表示也随之确定，如图 2.3 所示。这种表示方法被称之为 Tanner 图[38]，其中每一个方形节点对应一个校验方程，被称为校验节点；每一个圆形节点对应码字中的一个比特，被称为变量节点。例如校验方程$\hat{u}_1+\hat{u}_3+\hat{u}_4=0$对应在图中则是$v_1,v_3,v_4$与$c_1$相连。Tanner 图$G$也可描述为$G=(\mathbb{C}\cup\mathbb{V},\mathbb{E})$，其中$\mathbb{C},\mathbb{V},\mathbb{E}$分别代表校验节点集合、变量节点集合、边集合。

2.3 信念传播算法及陷阱集理论

信念传播 (belief-propagation, BP) 算法是一类通用的线性分组码解码算法。目前应用最广泛的 LDPC 码常常使用 BP 类算法及其各种变种进行解码。给定输入向量$\mathbf{r}$、最大迭代次数I_{max}、噪声标准差σ，作为最基础的 BP 类算法，对数域的 sum-product 算法的流程如下：

（1）初始化：

$$l_{v_i}^{ch},l_{v_i\to c_j}^{0}=\ln\frac{\Pr(y_i=+1|r_i)}{\Pr(y_i=-1|r_i)}=\frac{2r_i}{\sigma^2} \tag{2.11}$$

（2）校验节点向变量节点传递信息，其中$N(c_j)\backslash\{v_i\}$代表 Tanner 图中除去v_i后节点c_j的相邻节点集合，下同：

$$l_{c_j\to v_i}^{k+1}=2\tanh^{-1}\left(\prod_{v_{i'}\in N(c_j)\backslash\{v_i\}}\tanh(\frac{l_{v_{i'}\to c_j}^{k}}{2})\right) \tag{2.12}$$

（3）变量节点向校验节点传递信息：

$$l_{v_i\to c_j}^{k+1}=l_{v_i}^{ch}+\sum_{c_{j'}\in N(v_i)\backslash\{c_j\}}l_{c_{j'}\to v_i}^{k+1} \tag{2.13}$$

（4）输出：

$$l_{v_i}^{k+1}=l_{v_i}^{ch}+\sum_{c_{j'}\in N(v_i)}l_{c_{j'}\to v_i}^{k+1} \tag{2.14}$$

$$\hat{u}_i^{k+1}=\begin{cases}0,l_{v_i}^{k+1}\geq 0\\1,l_{v_i}^{k+1}<0\end{cases} \tag{2.15}$$

该算法中步骤 2、3、4 会迭代多次直到迭代次数达到I_{max}或者输出满足$\hat{\mathbf{u}}\mathbf{H}^{\mathrm{T}}=0$。BP 类算法能收敛到最优解的前提是 Tanner 图中没有环。校验节点c_j传播至变量节点v_i的信息$l_{c_j\to v_i}^{k+1}$的物理意义为校验节点c_j所对应的校验方程被满足的情况下变量节点v_i对应的 LLR，证明如下：

考虑单个校验节点c有$m+1$个相邻变量节点$v_1,v_2,\dots v_{m+1}$。设除去其中v_i后的$m=|N(c)\backslash\{v_i\}|$个变量节点为$N'(c)=\{v_1',v_2',\dots,v_m'\}$，若$N'(c)$中节点的校验和为 1

则$v_i=1$才能满足校验方程，反之则需要$v_i=0$。以此类推，若校验节点被满足，则$v_i=0$的概率为$N'(c)$中有偶数个节点为 1 且其他节点皆为 0 的概率，$v_i=1$的概率等于$N'(c)$中有奇数个节点为 1 且其他节点皆为 0 的概率。定义T_P为节点集合P中的所有元素取 1 且节点集合$N'(c)\backslash P$中的所有元素取 0 的概率。由于假设各节点之间概率独立，则有$T_P=\prod_{v_i\in P}\Pr(v_i=1|r_i)\cdot\prod_{v_j\in N'(c)\backslash P}\Pr(v_j=0|r_i)$。那么节点$v_i$的概率分布可以用其他$m$个节点的概率分布表示为：

$$\begin{aligned}\Pr(v_i=0|r_i)=&\underbrace{T_{\{\emptyset\}}}_{C_m^0\text{项}}+\underbrace{T_{\{v_1',v_2'\}}+T_{\{v_1',v_3'\}}+\cdots T_{\{v_{m-1}',v_m'\}}}_{C_m^2\text{项}}\\&+\underbrace{T_{\{v_1',v_2',v_3',v_4'\}}+T_{\{v_1',v_2',v_3',v_5'\}}+\cdots}_{C_m^4\text{项}}+\cdots+\underbrace{T_{\{v_1',v_{2\cdots}'\}}+\cdots}_{C_m^{2\lfloor\frac{m}{2}\rfloor}\text{项}}\end{aligned}\tag{2.16}$$

$$\begin{aligned}\Pr(v_i=1|r_i)=&\underbrace{T_{\{v_1'\}}+T_{\{v_2'\}}+\cdots T_{\{v_m'\}}}_{C_m^1\text{项}}+\underbrace{T_{\{v_1',v_2',v_3'\}}+T_{\{v_1',v_2',v_4'\}}+\cdots T_{\{v_{m-2}',v_{m-1}',v_m'\}}}_{C_m^3\text{项}}\\&+\underbrace{T_{\{v_1',v_2',v_3',v_4',v_5'\}}+T_{\{v_1',v_2',v_3',v_4',v_6'\}}+\cdots}_{C_m^5\text{项}}+\underbrace{T_{\{v_1',v_{2\cdots}'\}}+\cdots}_{C_m^{2\lfloor\frac{m+1}{2}\rfloor-1}\text{项}}\end{aligned}\tag{2.17}$$

对任意一个节点的 1/2 的 LLR 值取$\tanh(\cdot)$操作可得到其为 0 和 1 的概率之差：

$$\tanh\left(\frac{l_v}{2}\right)=\frac{e^{l_v}-1}{e^{l_v}+1}=\Pr(v=0|r)-\Pr(v=1|r)\tag{2.18}$$

由(2.18)，有：

$$\prod_{v_j\in N(c)\backslash\{v_i\}}\tanh(\frac{l_{v_{i'}}}{2})=\prod_{v_x\in N(c)\backslash\{v_i\}}\Pr(v_x=0|r_x)-\Pr(v_x=1|r_x)\tag{2.19}$$

使用二项式定理将公式(2.19)右侧展开后共有2^m项，其中有C_m^0项的因子只包含$\Pr(v_x=0|r_x)$，对应$T_{\{\emptyset\}}$；有C_m^1项的因子中只有一个$\Pr(v_x=1|r_x)$而其他因子是$\Pr(v_x=0|r_x)$，对应$T_{\{v_1'\}}+T_{\{v_2'\}}+\cdots T_{\{v_m'\}}$，且符号为负号。以此类推，对于每一项而言，若其因子中有偶数个取$\Pr(v_x=1|r_x)$则符号为正，有奇数个取$\Pr(v_x=1|r_x)$则符号为负。使用(2.16)(2.17)分别代换(2.19)展开后的正项和负项，有：

$$\begin{aligned}\prod_{v_{i'}\in N(c)\backslash\{v_i\}}\tanh(\frac{l_{v_{i'}}}{2})=&\underbrace{T_{\{\emptyset\}}}_{C_m^0\text{项}}+\cdots+\underbrace{T_{\{v_1',v_{2\cdots}'\}}+\cdots}_{C_m^{2\lfloor\frac{m}{2}\rfloor}\text{项}}\\&-(\underbrace{T_{\{v_1'\}}+T_{\{v_2'\}}+\cdots T_{\{v_m'\}}}_{C_m^1\text{项}}+\cdots+\underbrace{T_{\{v_1',v_{2\cdots}'\}}+\cdots}_{C_m^{2\lfloor\frac{m+1}{2}\rfloor-1}\text{项}})\\=&\Pr(v_i=0|r_i)-\Pr(v_i=1|r_i)\\=&\tanh\left(\frac{l_{v_i}}{2}\right)\end{aligned}\tag{2.20}$$

两边同时取$\tanh^{-1}(\cdot)$后乘 2得到公式(2.21)，与公式(2.12)的形式相同。

$$l_{v_i} = 2\tanh^{-1}\left(\prod_{v_{i'}\in N(c_j)\backslash\{v_i\}} \tanh(\frac{l^k_{v_{i'}}}{2})\right) \tag{2.21}$$

由于在公式(2.12)(2.13)分别使用了连乘和加法操作，所以该算法也叫作和-积(sum-product) 算法。Min-sum 算法是 BP 算法的一个变种，可以视为 sum-product 算法的近似版本。在 min-sum 算法中公式(2.11)可近似为：

$$l^{ch}_{v_i}, l^0_{v_i\to c_j} = r_i \tag{2.22}$$

公式(2.12)可以近似为：

$$l^{k+1}_{c_j\to v_i} = \left(\prod_{i'\in N(c_j)\backslash\{v_i\}} \text{sign}\left(l^k_{v_{i'}\to c_j}\right)\right)\cdot \min_{v_{i'}\in N(c_j)\backslash\{v_i\}} \left|l^k_{v_{i'}\to c_j}\right| \tag{2.23}$$

对 LLR 取$\tanh(\cdot)$后的值处于$(-1,+1)$之间，因此简单地将$\tanh(\cdot)$的连乘替换为取最小值会导致幅度偏大。通过在(2.23)的基础上添加一个处于(0,1)之间的收缩系数可以解决这个问题，这种方法叫做 attenuated min-sum 算法：

$$l^{k+1}_{c_j\to v_i} = \left(\prod_{i'\in N(c_j)\backslash\{v_i\}} \text{sign}\left(l^k_{v_{i'}\to c_j}\right)\right)\cdot \alpha \min_{v_{i'}\in N(c_j)\backslash\{v_i\}} \left|l^k_{v_{i'}\to c_j}\right| \tag{2.24}$$

在 attenuated min-sum 算法的基础上还可以设置一个阈值用来控制是否对 LLR 幅度进行收缩，这种方法叫做 threshold attenuated min-sum（TAMS）算法[39]：

$$\begin{cases} l^{k+1}_{c_j\to v_i} = \left(\prod_{i'\in N(c_j)\backslash\{v_i\}} \text{sign}\left(l^k_{v_{i'}\to c_j}\right)\right)\cdot \alpha \min_{v_{i'}\in N(c_j)\backslash\{v_i\}} \left|l^k_{v_{i'}\to c_j}\right| \\ \qquad \text{if } \min_{i'\in N(c_j)\backslash\{v_i\}} \left|l^k_{v_{i'}\to c_j}\right| > \tau \\ l^{k+1}_{c_j\to v_i} = \left(\prod_{i'\in N(c_j)\backslash\{v_i\}} \text{sign}\left(l^k_{v_{i'}\to c_j}\right)\right)\cdot \min_{v_{i'}\in N(c_j)\backslash\{v_i\}} \left|l^k_{v_{i'}\to c_j}\right| \\ \qquad \text{otherwise} \end{cases} \tag{2.25}$$

2.4 神经网络

神经网络可以被认为是一类从训练数据中自动构造函数的算法。其基本的原理是给定一个模型f、参数组$\mathbf{w}$、输入样本$\mathbf{r}$和标签样本$\mathbf{u}$，对参数组$\mathbf{w}$进行优化使得神经网络的输出$f(\mathbf{w},\mathbf{r})$尽量贴近样本标签$\mathbf{u}$。而衡量输出$f(\mathbf{w},\mathbf{r})$和标签差距的准则我们称之为损失函数。神经网络算法分两阶段：训练阶段和测试阶段。在训练阶段神经网络通过大量的输入和输出样本对参数$\mathbf{w}$进行调整，从而最小化损失函数。在测试阶段神经网络会使用训练阶段固定下来的参数组$\mathbf{w}$进行推理，这里的参数组$\mathbf{w}$被称为神经网络的权重。

设神经网络的L层每层权重为$w^1, w^2, \ldots w^L$，激活函数分别为$f^1, f^2, \ldots, f^L$，损

失函数为$C(x,y)$，则神经网络的输出为，

$$\hat{\mathbf{u}} = f(\mathbf{w},\mathbf{r}) = f^{L}\left(w^{L}, f^{L-1}\left(w^{L-1}, f^{L-1}\left(\cdots f^{1}(w^{1},\mathbf{r})\right)\right)\right), \tag{2.26}$$

为了使损失函数最小化，需要找到损失函数的极值点，也就是使得$\nabla C = 0$成立的$\mathbf{w}$。而由于神经网络的非线性且神经网络常常拥有大量参数，通常无法直接求解其极值点。在实际情况中，人们常常使用基于随机梯度下降的方法 (stochastic gradient descent, SGD)，即通过迭代的方式多次更新权重。SGD 类方法在每一次更新时尽量选择使损失函数值下降最快的方向，但是为了避免陷入局部最优，其计算出的更新方向除了梯度外往往会增加动量或随机扰动。在每次迭代更新时都需要随机采样一批样本，我们可以认为其分布与总体的真实分布一致。对于一次更新而言，样本是固定的。因此在这种情况下损失函数可以视为只关于权重$\mathbf{w}$的函数$C(\mathbf{w})$。计算样本对应的$\nabla C(\mathbf{w})$，并使用$\mathbf{w} = \mathbf{w} - \eta\nabla C(\mathbf{w})$进行更新。其中$\eta$被称为学习率，用于控制每一次更新的步长。通常来讲更小的学习率会使得网络收敛更慢，而更大的学习率会使得网络收敛更快但是在极值点附近会引起震荡。

2.5 本章小结

本章首先介绍了通信所使用的模型以及编码的基本知识；随后描述了 QR 码、BCH 码、punctured RM 码的编码方法；之后又阐述了这几种码的生成矩阵以及校验矩阵的构建。此外，本章还介绍了 BP 类算法，包括对数域的 sum-product 算法以及简化后的 min-sum、attenuated min-sum 和 threshold attenuated min-sum 算法。最后本章简要介绍了神经网络的原理。

3 基于 BP 类算法的神经网络解码器

由于 BP 类算法具有较强的假设前提（各节点概率相互独立），其在有较多短环或小陷阱集的 Tanner 图上性能会发生大幅度劣化。对于校验矩阵密度较高的循环码尤其如此，校验矩阵密度的增高使得短环数和小陷阱集的数量也快速增加，这进一步影响了 BP 类算法的性能。本章首先对影响 BP 类算法性能的因素进行分析。在此基础上，我们详细介绍基于 BP 类算法的神经网络解码器并进行多种改进，在提高解码性能的同时降低复杂度。

3.1 影响 BP 类算法性能的因素分析

BP 类算法被证明在无环图上可以收敛到真实的概率分布，而在有环图上则无法保证。即使如此，实践中在有环图上使用 BP 类算法依然有机会收敛到我们所期望的结果。在有环图上运行 BP 类算法时，Tanner 图中短环的数目被认为是影响 BP 类算法性能的重要因素之一。一般地，Tanner 图中的短环数量越少，BP 类算法解码的准确率就越高。在构造 LDPC 码的校验矩阵时，人们需要通过各种方式尽量使得 Tanner 图中的短环数量最少。

算法 3.1: 环搜索算法[41]

输入： Tanner 图: $G = (\mathbb{C} \cup \mathbb{V}, \mathbb{E})$，环长: g_l

输出： 所有长度为g_l的环型路径集合：P_{g_l}

$P_{g_l} = \emptyset$

For v **in** $\mathbb{V}$ **do**

  $C_e \leftarrow N(v)$ //寻找v的所有相邻节点

  If $|C_e| = 0$ **then**

    Continue

  Else

    For c **in** C_e **do**

      $P_{g_l} \leftarrow P_{g_l} \cup$AllSimplePath$(G, c, v, g_l - 1)$

      //寻找图G中所有以c为起点，v为终点，长度为$g_l - 1$的简单路径

    End for

    Remove(G, v) //从图G中删除v以及v相连的边

  End if

End for

Return P_{g_l}

Tanner 图属于二部图，任意一个变量节点的相邻节点都是校验节点，反之，任意一个校验节点的相邻节点也都是变量节点。所以对于任意一个 Tanner 图，其环长只可能是不小于 4 的偶数。在设计 LDPC 码时为了提升 BP 算法解码的准确率，常常要避免 Tanner 图中 4 环和 6 环的出现。但是由于大部分循环码的校验矩阵中 1 的数量要远远多于 LDPC，所以其 Tanner 图中难以避免地存在大量 4 环。环的搜索算法如算法 3.1 所示。在 Tanner 图中寻找长度为g_l的短环时，只需遍历每一个变量节点，并寻找该变量节点到与之相邻的校验节点之间长度为g_l-1的简单路径（即没有环路的路径）。找到所有这样的路径后便可以将该节点以及所有与之相邻的边都删去，因为包含该节点的环路都已经被穷举。这样做可以加快接下来的搜索过程。

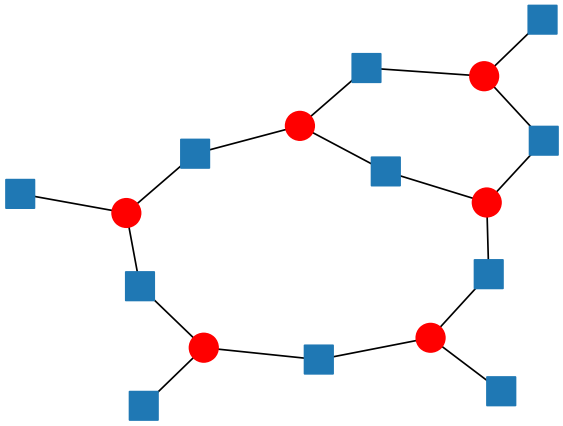

图 3.1 (6, 4)基本陷阱集

Fig. 3.1 A (6, 4) elementary trapping set.

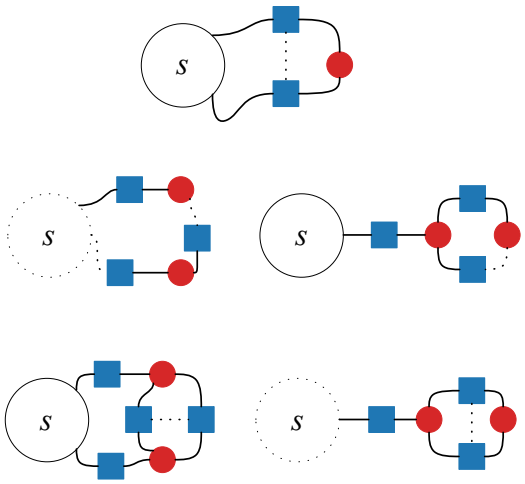

图 3.2 可能的陷阱集扩展路径

Fig. 3.2 Possible expansions of a trapping set.

除了短环以外，陷阱集[40]也是影响 BP 类算法性能的重要因素之一。这类结构会导致 BP 类算法在某些节点解码失败。给定一个 Tanner 图$G=(\mathbb{C}\cup\mathbb{V},\mathbb{E})$以及变量节点集合$\mathbb{S}\subset\mathbb{V}$，定义$\mathbb{S}$在$G$中的邻接节点集合为$\Gamma(\mathbb{S})$。由于$G$是二部图，所以$\Gamma(\mathbb{S})\subset\mathbb{V}$。定义$\mathbb{S}$的生成子图$T(\mathbb{S})$为$(\mathbb{S}\cup\Gamma(\mathbb{S}),\{(u,v)\in\mathbb{E},u\in\mathbb{S},v\in\Gamma(\mathbb{S})\})$。将$\Gamma(\mathbb{S})$中度数为奇数的节点集合记为$\Gamma_o(\mathbb{S})$，称为非满足校验节点。若$|\mathbb{S}|=a,|\Gamma_o(\mathbb{S})|=b$，那么$T(\mathbb{S})$被称为一个$(a,b)$陷阱集。此外，若$\Gamma(\mathbb{S})$中的节点度数只有 1 和 2 两种，那么$T(\mathbb{S})$被称为一个基本陷阱集。基本陷阱集被认为对正确解码的危害更大。BP 类算法每进行一次迭代时，非满足校验节点会更正与其相邻的变量节点从而使校验方程得到满足。而陷阱集中被虚假满足的校验节点则会增强与之相邻的变量节点的置信度从而使得其中发生的错误更难被纠正。因此当陷阱集中的非满足校验节点占据优势时解码更容易成功，反之则更容易出错。此外，陷阱集中的变量节点越多则越不容易发生校验节点被虚假满足的现象。综上，陷阱集的a,b值越大则对解码负面影响越小，反之则影响越大。图 3.1 展示了一个典型的（6，4）基本陷阱集，其中红色圆形和蓝色方形节点分别代表变量节点和校验节点。基本陷阱集的搜索算法[41]6942-6958 如算法 3.2 所示。使用该算法穷举陷阱集时需要运行多次，不断将小的陷阱集扩充为更大的陷阱集从而完成整个陷阱集的搜索。此外，在初次运行该算法时需要输入 Tanner 图中的环路径，这可以通过算法 3.1 得到。在每次扩充时该算法会检查可用的“路径”和“棒棒糖路径”（如图 3.2 所示，其中s代表扩充前的陷阱集，(a)(b)(d)对应“路径”，(c)(e)对应“棒棒糖路径”），并在每次扩充时保证这次扩充是最小的，防止陷阱集的重复或者遗漏。

边上消息传递的幅度同样也是影响 BP 类算法性能的一个重要因素。因为 BP 类算法的最优性只有在无环图中才能得到保证，在这种情况下可以认为变量节点取 0/1 的概率是相互独立的。而实际中这个假设并不成立，所以对于边上传递的 LLR 信息幅度需要进行加权。而 min-sum 算法更是如此，除了不满足节点间概率相互独立这一假设，min-sum 算法还将 sum-product 算法中$\tanh(\cdot)$值的连乘简化为取最小值。我们知道$\tanh(\cdot)$函数的值域在$(-1,+1)$之间，因此将连乘用取最小值替换会使得传播的消息幅度偏高，这种现象在节点度数较高的循环码上更为明显。在实际中，人们使用 attenuated min-sum 算法（参见公式(2.24)）来缓解这一问题，其中的收缩系数常常是依经验而定，且在迭代过程中不会变化。为每次迭代分配不同的收缩系数可以提升性能，但迭代次数较多时为每一次迭代依次确定收缩系数的工作量较大。因此训练神经网络自动化确定收缩系数可能是效率更高的做法，且由于更多权重的引入，神经网络可以更精细地调整每一层中不同边的权重值，从而获得更高的准确率。

算法 3.2: 基本陷阱集搜索算法[41]6942-6958

输入：Tanner 图: $G=(\mathbb{C}\cup\mathbb{V},\mathbb{E})$，小陷阱集的集合$L_{in}$（其中每一个元素都是一个陷阱集的变量节点集合），目标陷阱集的最大变量节点个数k和最大奇数校验节点数T

输出：更大的陷阱集集合：L_{out}

$L_{out}=\emptyset$

For p **in** L_{in} **do**

$G'\leftarrow$Remove$(G,\Gamma_e(p)\cup N(\Gamma_e(p)))$

//从G中删除$\Gamma_e(p)$及其所有相邻的节点构建新图

$i_{max}\leftarrow(k-|p|),\mathcal{G}\leftarrow\emptyset$

For c **in** $\Gamma_o(p)$ **do**

$\mathcal{P},i_{\mathcal{P}}\leftarrow$FindPaths$(G',c,\Gamma_o(p)\backslash\{c\},i_{max})$

//寻找所有起点为c，终点为$\Gamma_o(p)\backslash\{c\}$中元素，长度不超过$i_{max}$的路径$\mathcal{P}$

//（$\mathcal{P}$中每条路径只需记录变量节点）。以及$\mathcal{P}$中路径的最短值$i_{\mathcal{P}}$

$\mathcal{L},i_{\mathcal{L}}\leftarrow$FindLollipops$(G',c,i_{max})$

//寻找以c为起点和终点且长度不超过i_{max}的棒棒糖路径上的变量节点

//$\mathcal{L}$以及路径的最短值$i_{\mathcal{L}}$

$i=\min(i_{\mathcal{P}},i_{\mathcal{L}})$

$\mathcal{G}_c\leftarrow$FilterPaths$(\mathcal{P}\cup\mathcal{L},i)$//寻找$\mathcal{P}$和$\mathcal{L}$中长度最小的路径

If $i<i_{max}$ **then**

$i_{max}\leftarrow i$

$\mathcal{G}\leftarrow\mathcal{G}_c$

Else

$\mathcal{G}\leftarrow\mathcal{G}\cup\mathcal{G}_c$

End if

End for

For s **in** $\mathcal{G}$ **do**

$p'\leftarrow p\cup s$

If $t'\notin L_{out}$ **and** $|\Gamma_o(t')|\leq T$ **then**

$L_{out}\leftarrow L_{out}\cup\{t'\}$

End if

End for

Remove(G,v) //从图G中删除v以及与v相连的边

End for

Return L_{out}

3.2 神经网络 BP 算法

为了缓解环对 BP 类算法的性能影响，研究人员们提出了神经网络 BP 算法。其设想是通过对 Tanner 图上的每条边赋予独立权重，在环上传递的信息一部分被放大，一部分被缩小，从而减轻环对 BP 类算法的影响。给定一个如图 3.3 所示的 Tanner 图，其对应的神经网络 BP 解码器结构如图 3.4 所示。图中包含了两次 BP 迭代，每一次 BP 迭代对应的神经网络层使用黑色虚线框标记。若想构建对应I_{max}次迭代的神经网络，只需重复第二个虚线框的内容，即$\mathrm{IAB}\underbrace{\mathrm{A'BA'B}\ldots}_{2(I_{max}-1)}\mathrm{O}$。I 层和 O 层分别为输入和输出层，其中的每一个神经元都对应一个比特的 LLR。虚线框内的$\mathrm{A}, \mathrm{A}', \mathrm{B}$层为隐藏层，其中的每一个神经元都对应 Tanner 图中的一条边。

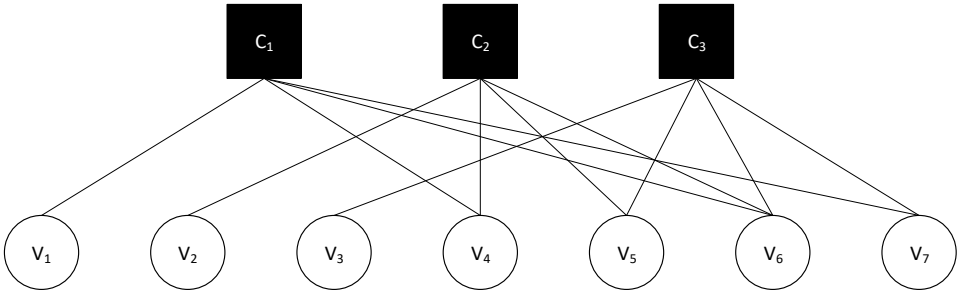

图 3.3 (7, 4)汉明码对应的 Tanner 图

Fig. 3.3 Tanner graph of the (7,4) Hamming code.

神经网络 BP 算法中每层之间的神经元连接构造规则如下：

- I 层到 A 层：若 A 层中某神经元对应在 Tanner 图中的边与 I 层中某神经元对应在 Tanner 图中的变量节点相连，则在神经网络中添加这两个神经元的连接。
- A/A’层到 B 层：若 A 层中某神经元对应在 Tanner 图中的边与 B 层中某神经元对应在 Tanner 图中的边连接在同一个校验节点上，且两个神经元对应的不是 Tanner 图中的同一条边，则在神经网络中添加这两个神经元的连接。
- B 层到 A’层：若 B 层中某神经元对应在 Tanner 图中的边与 A’层中某神经元对应在 Tanner 图中的边连接在同一个变量节点上，且两个神经元对应的不是 Tanner 图中的同一条边，则在神经网络中添加这两个神经元的连接。图 5 中的 A’层神经元前有黑色方块，对应公式(2.11)中的信道 LLR。
- B 层到 O 层：若 B 层中某神经元对应在 Tanner 图中的边与 O 层中某神经元对应在 Tanner 图中的变量节点相连，则在神经网络中添加这两

个神经元的连接。图 5 中的 O 层神经元前有黑色方块，对应公式(2.11)中的信道 LLR。

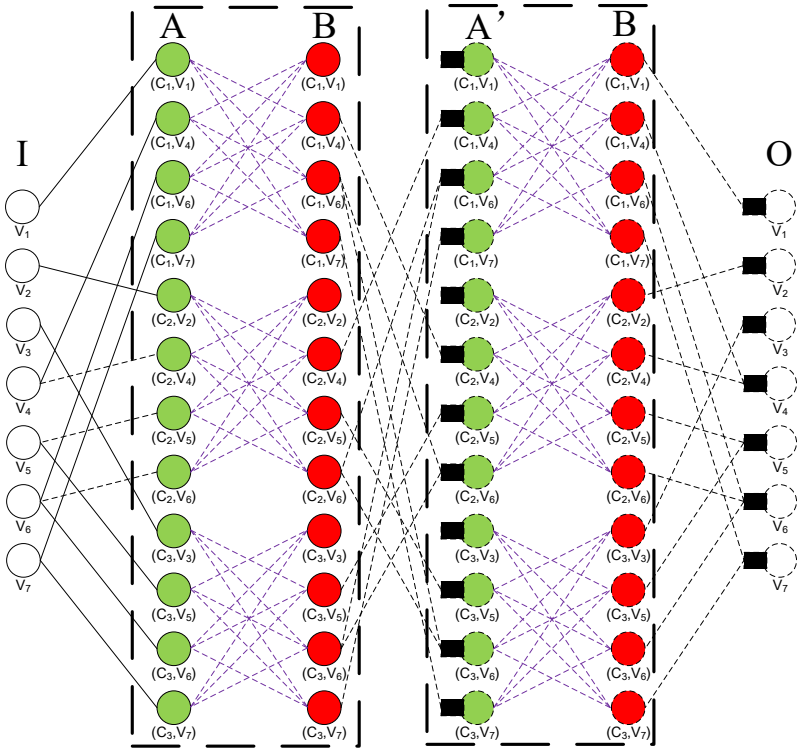

图 3.4 (7, 4)汉明码对应神经网络 BP 解码器结构图

Fig. 3.4 The structure of the neural BP decoder corresponding to the (7, 4) Hamming code.

在 Tanner 图上为每条边添加权重后，神经网络 sum-product 算法的迭代公式从表示原 sum-product 算法的公式(2.12)(2.13)分别变为：

$$l_{c_j\to v_i}^{k+1}=2\tanh^{-1}\left(\prod_{v_i'\in N(c_j)\backslash\{v_i\}}\tanh(\alpha_{i'\to j}\cdot\frac{l_{v_{i'}\to c_j}^{k}}{2})\right), \tag{3.1}$$

$$l_{v_i\to c_j}^{k+1}=l_{v_i}^{ch}+\sum_{c_{j'}\in N(v_i)\backslash\{c_j\}}\beta_{j'\to i}\cdot l_{c_{j'}\to v_i}^{k+1}. \tag{3.2}$$

与神经网络 BP 类似，神经网络 min-sum 中对应公式(2.23)的步骤变为：

$$l_{c_j\to v_i}^{k+1}=\left(\prod_{i'\in N(c_j)\backslash\{v_i\}}\operatorname{sign}\left(l_{v_{i'}\to c_j}^{k}\right)\right)\cdot\min_{v_i'\in N(c_j)\backslash\{v_i\}}\left|\alpha_{i'\to j}\cdot l_{v_{i'}\to c_j}^{k}\right|. \tag{3.3}$$

公式(2.25)中的 TAMS 算法推广到神经网络中则变为：

$$\begin{cases} l_{c_j\to v_i}^{k+1} = \alpha' \left(\prod_{i'\in N(c_j)\setminus\{v_i\}} \operatorname{sign}\left(l_{v_{i'}\to c_j}^{k}\right) \right) \cdot \min_{v_i'\in N(c_j)\setminus\{v_i\}} \left|\alpha_{i'\to j}\cdot l_{v_{i'}\to c_j}^{k}\right| \\ \qquad \text{if } \min_{v_i'\in N(c_j)\setminus\{v_i\}} \left|\alpha_{i'\to j}\cdot l_{v_{i'}\to c_j}^{k}\right| > \tau \\ l_{c_j\to v_i}^{k+1} = \left(\prod_{i'\in N(c_j)\setminus\{v_i\}} \operatorname{sign}\left(l_{v_{i'}\to c_j}^{k}\right) \right) \cdot \min_{v_i'\in N(c_j)\setminus\{v_i\}} \left|\alpha_{i'\to j}\cdot l_{v_{i'}\to c_j}^{k}\right| \\ \qquad \text{otherwise} \end{cases} . \tag{3.4}$$

上述两种神经网络 min-sum 解码器的变量节点更新步骤与神经网络 sum-product 一样变为(3.2)。公式(3.1-3.4)中出现的$\alpha_{i'\to j}, \beta_{j'\to i}$均为网络权重，其取值范围为(0,1)。

我们使用二进制交叉熵 (binary cross-entropy, BCE) 损失函数来衡量输出 LLR 与真实码字$\mathbf{u}$之间的差距，假设神经网络的输出 LLR 为$\mathbf{l}^o$，sigmoid 函数为$\sigma(\cdot)$，BCE 损失函数如公式(3.5)所示。

$$L(\mathbf{l}^o, \mathbf{u}) = -\frac{1}{n}\sum_{i=1}^{n}\left(u_i \ln \sigma(-l_i^o) + (1-u_i)\ln\left(\sigma(l_i^o)\right)\right) \tag{3.5}$$

其中$\sigma(\cdot)$为 sigmoid 函数，定义如下：

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.6}$$

由于 BP 类算法在图上传递和输出的结果都可视为 LLR，如(3.7)所示，使用 sigmoid 函数便可将 LLR 转化为概率，因此使用 BCE 函数衡量损失与 BCE 原本的定义相符。

$$\begin{aligned} \sigma\left(\ln\frac{\Pr(s_i=0|r_i)}{\Pr(s_i=1|r_i)}\right) &= \frac{1}{1+\dfrac{\Pr(s_i=1|r_i)}{\Pr(s_i=0|r_i)}} \\ &= \Pr(s_i=0|r_i) \end{aligned} \tag{3.7}$$

针对部分码字该函数也可改为多损失函数的形式，即在网络计算完一次 BP 迭代时将中间结果作为输出计入总损失函数中，这样做的好处是可以加速网络收敛速度。但是若将多损失函数应用于迭代次数较少时无法较好解码的码字，则会影响网络收敛。多损失交叉熵函数具体如下：

$$L_M(\mathbf{l}^1, \ldots, \mathbf{l}^{I_{max}}, \mathbf{u}) = -\frac{1}{I_{max}\cdot n}\sum_{j=1}^{I_{max}}\sum_{i=1}^{n}\left(u_i \ln\sigma\left(-l_i^j\right) + (1-u_i)\ln\left(\sigma\left(l_i^j\right)\right)\right). \tag{3.8}$$

为了验证神经网络 BP 算法缓解短环和陷阱集效应的有效性，我们通过实验仿真进行测试。具体的方法是：使用神经网络 sum-product 解码器对带噪声的数据进行解码并统计出错位置，当收集到足够多的错误时停止。以 QR (47, 24) 码为例，统计结果如图 3.5 所示。图中使用了 5 次迭代，在$\frac{E_b}{N_0} = 8$ dB 的情况下收集了超过 5 000 次比特错误。可以看到使用 sum-product 解码器进行解码，在四环数比较多

的第 10~30 位节点，错误反而更少，而错误集中发生的 38、42、43、46 这几个节点处的四环数也并未明显高于其他节点。尽管通常而言使用 BP 类算法解码 LDPC 码时的性能受四环影响极其严重，但是图 3.5 表明，BP 类算法在解码 QR 码这类高密度码时，四环可能并非是主导错误发生的唯一因素。

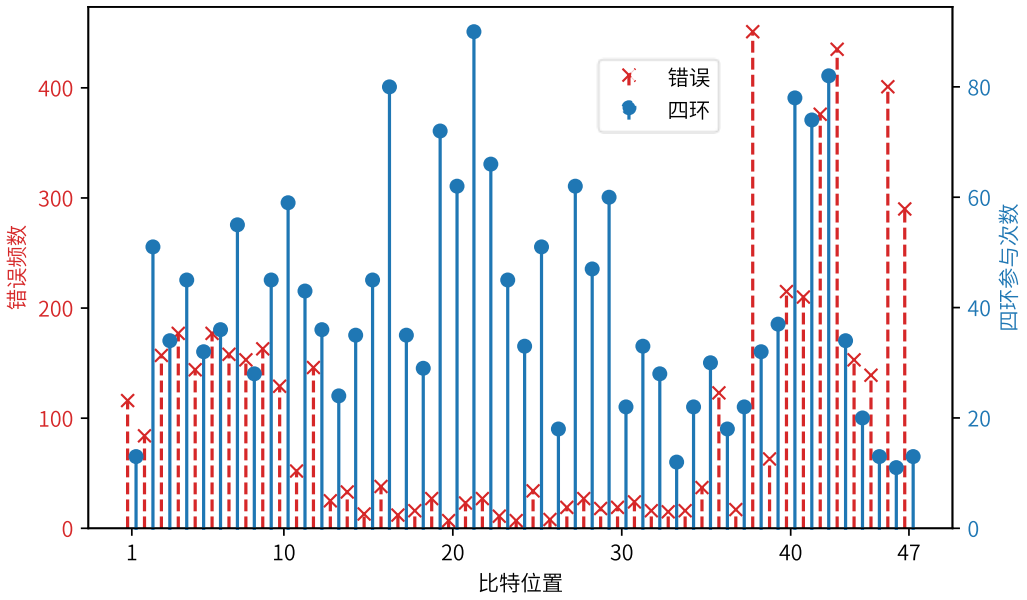

图 3.5 错误频数和四环分布图

Fig. 3.5 The distribution of error frequency and 4-cycles.

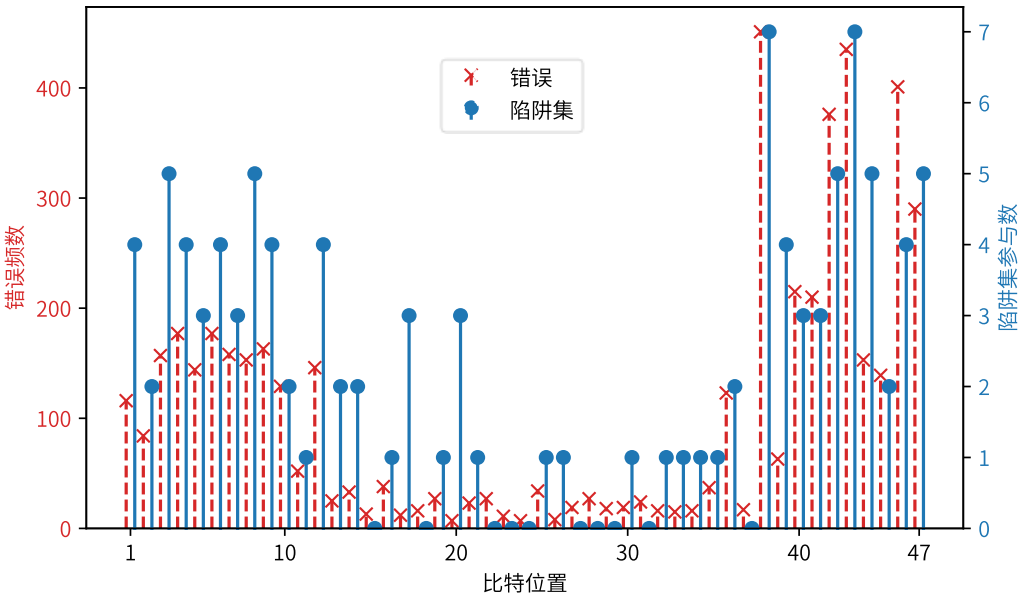

图 3.6 错误频数和小陷阱集分布图

Fig. 3.6 The distribution of error frequency and small trapping sets.

考虑上文提到的第二个因素，我们尝试考察小型基本陷阱集对于 BP 类算法解码高密度码的性能影响。图 3.6 展示了小型基本陷阱集和错误的分布，这里的“小型”指陷阱集的大小参数满足$a+b<7, a<4$。可以看到和四环数相比，小型陷阱集的分布与错误的分布更加吻合。因此我们可以推断在此种情况下，小型的基本陷阱集贡献了大部分错误，四环则并非发生错误的主要因素。图 3.6 中错误高发的 38、42 节点处于同一个 (2, 3) 陷阱集中（如图 3.7 所示），而 43、46 节点也处于同一个 (2, 3) 陷阱集中（如图 3.8 所示）。

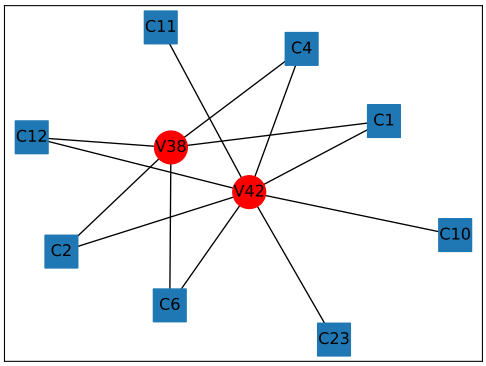

图 3.7 节点 38、42 对应的陷阱集

Fig. 3.7 A trapping set containing nodes 38 and 42.

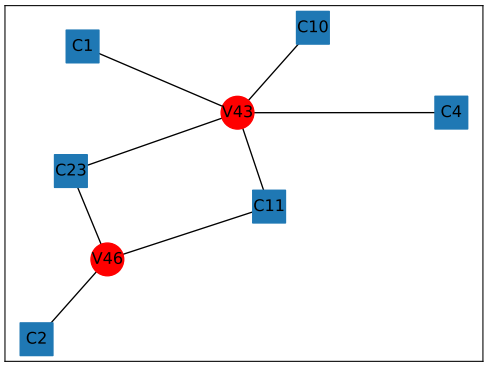

图 3.8 节点 43、46 对应的陷阱集

Fig. 3.8 A trapping set containing nodes 43 and 46.

3.3 神经网络循环 mRRD 算法

针对陷阱集导致的解码错误问题，本章提出基于 mRRD 算法和循环置换来构造新的解码算法。其中 mRRD 算法从码的置换群中随机挑选置换，并利用置换对接收到的信息进行多次重排和解码，最后从多个输出中挑选最可能的一个结果作为最终输出。由于进行置换后接收向量中的比特顺序会被打乱，在多次置换和解码后，原来处于陷阱集中的比特被移出陷阱集，陷阱集外的正确消息则会被传入陷阱集，从而有助于 Tanner 图中消息的正确收敛。置换的具体操作如下，若给定置换$\sigma=\begin{pmatrix}1 & 2 & 3\\ 3 & 2 & 1\end{pmatrix}$以及码字$\mathbf{u}=(u_1,u_2,u_3)$，那么置换后的结果为$\sigma(\mathbf{u})=(u_3,u_2,u_1)$，这里的置换$\sigma$使用的表示方法称为两行表示法，第一行为置换前的位置，第二行为置换后的位置。并不是任意一个置换都可以应用到 mRRD 算法中。在 mRRD 算法中使用的所有置换必须都在码的置换群中，码字$\mathcal{C}$的置换群$\mathrm{Per}(\mathcal{C})$是指由所有能够保持码字结构不变的置换构成的一个群，即：

$$\mathrm{Per}(\mathcal{C}) = \{\sigma | \sigma(\mathbf{u}) \in \mathcal{C}, \forall \mathbf{u} \in \mathcal{C}\} \tag{3.9}$$

由于置换群中的置换可以保证码字在被置换后仍然属于码字集合，所以对接收向量进行此种置换可以保证被置换后的向量送入 BP 类解码器所产生的结果是有效的。事实上，将接收向量进行置换再并基于原校验矩阵解码等价于在接收向量不变的情况下对校验矩阵做变换然后解码，两种方法均不影响解码结果的有效性。只是对接收向量进行置换并解码后需要将输出的解码结果进行逆置换才可以达到期望的效果。“LMS (least metric selector)”是一种判决准则，其作用是在多个 mRRD 解码后的候选码字中选择一个最接近于输入向量的码字作为输出，这样选出的码字是候选码字集合中最优的结果。若输入 LMS 的候选解码结果集合为$\hat{\mathbb{U}}$，则输出结果如下：

$$LMS: \hat{\mathbf{u}} = \mathrm{argmax}_{\hat{\mathbf{u}} \in \hat{\mathbb{U}}} \sum_{i=1}^{n} |r_i - \hat{u}_i|^2 \tag{3.10}$$

mRRD 算法可以很好地缓解陷阱集导致的错误收敛。但是对于某些码而言，其置换群较难求解或十分庞大，为了储存置换群需要消耗极高的存储空间。表 3.1 列举了一些常见 QR 码和 BCH 码的置换群大小。以 QR (23, 12) 为例，若使用前文提到的两行置换表示方法，由于第一行始终为 1 到 23 的顺序排列，可以省去，那么每一个置换需要存储 23 个整数。1 到 23 的整数可以用 5 个 bit 表示，则储存 QR (23, 12) 码的整个置换群需要$10\,200\,960 \times 5\ \mathrm{bit} \approx 6\,226\ \mathrm{KiB}$。同样的，对于 BCH (63, 57) 码则可能需要$2 \times 10^{10} \times 6\ \mathrm{bit} \approx 14305\ \mathrm{MiB}$。若只储存置换群的生成元而不储存整个置换群则可以极大程度地节省空间，因为置换群的生成元常常不超过 10 个。但是使用生成元生成整个置换群的 Schreier–Sims 算法[42]十分复杂，不便于硬件实现。

表 3.1. 几种典型 QR 码和 BCH 码的置换群大小

Table 3.1 The cardinality of permutation groups for some QR codes and BCH codes.

$\mathcal{C}$	QR (23, 12)	QR (89, 45)	QR (97, 49)	BCH (63, 45)	BCH (63, 57)
$\lvert\mathrm{Per}(\mathcal{C})\rvert$	10 200 960	3 916	4 656	10 584	$\approx 2\times10^{10}$

除此之外，给定一个置换群后在硬件上实现其中的任意置换也较为复杂。这是因为在使用随机置换后，某个位置上的比特可能会被置换到任意一个位置上，而且比特之间的顺序也可能被完全打乱。这就使得实现任意置换的电路变得十分复杂。为了解决以上两个问题，我们提出在 mRRD 规模较小时使用循环移位来代替随机置换。这样做有两个好处。首先是解决了置换群的求解和存储问题，循环码的置换群中一定包含循环移位，即对任意一个码字进行循环移位操作所产生的

结果仍然是一个码字。因此所有的循环移位构成的群是置换群的子群，这就保证了使用循环移位代替置换群所产生的结果的有效性；此外由于所有的循环移位都可以用一个整数来表示，这样就无需额外的储存空间，只需任意取一个整数并对输入进行整数次循环移位操作即可。最后，使用循环移位也解决了硬件实现的困难，因为在循环移位操作中每一位只会被置换到与其相邻的位置上，比特之间的顺序不会被打乱，从而降低电路实现的复杂度。循环移位和任意置换的对比如图 3.9 所示。

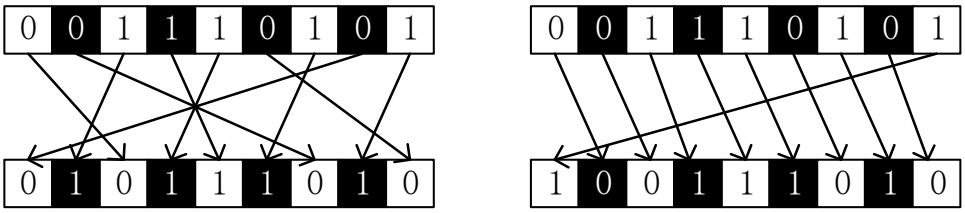

图 3.9 循环置换和随机置换的对比图

Fig. 3.9 Comparison between cyclic permutation and random permutation.

算法 3.3: 最佳移位算法

输入： 小陷阱集分布: D_T

输出： 最佳移位（右移）：σ

For $i \leftarrow 1$ **to** n **do**

$x \leftarrow D_T - (D_T >> n)$ //衡量移位后陷阱集的变化

$Score[i] \leftarrow \sum_{x_i>0} x_i$

End for

$S \leftarrow \{i | Socre[i] == \max(Score)\}$

If $|S| \neq 1$ **then**

$\sigma \leftarrow \text{argmax}_{\sigma \in S}(|\sigma - \frac{n}{2}|)$

Else

$\sigma \leftarrow S[1]$

结合前文中对陷阱集的分析，我们可以对循环移位做进一步挑选，找出对结果改善更明显的移位操作。在选出效果最好的移位后，我们通过反复使用这一种循环移位可以进一步降低实现复杂度。而且我们可以通过提前对校验矩阵进行分析从而固定一个适合的移位，这样就不会给解码过程带来额外的开销。考虑某码长为n的码，其小陷阱集分布为$D_T = (t_1, t_2, \ldots, t_n)$，其中$t_i$是变量节点$V_i$所参与的

小陷阱集总数。前文的分析中提到了 mRRD 可以通过置换将陷阱集中的比特移出陷阱集并将外部正确的信息传入陷阱集，所以我们可以根据陷阱集分布挑选出一个固定的循环移位，从而将参与陷阱集较多的节点移动到陷阱集较少的位置上，这样就可以增强 mRRD 算法对抗陷阱集的效果。具体的挑选算法如算法 3.3 中所述，该算法给n个循环移位分别按照陷阱集分布进行评分，然后从中选择评分最高的一个循环移位输出。若该算法发现有多个循环移位评分相同，则会从中选出长度最小的一个循环移位（也就是最接近 0 或n的）。假如算法对于 QR (47, 24) 输出了右移 40 位和右移 23 位，说明从陷阱集分布的角度看这两者都是最优解。但是右移 40 位等价于左移 7 位，而右移 23 位等价于左移 24 位。可以看到右移 40 位在化简后所需的位数比 23 或者 24 位都要少，所以最终的输出结果是右移 40 位（左移 7 位）。

算法 3.4: 循环 mRRD

输入: 最佳右移: σ, 校验矩阵: H 码长: n, 接收向量: $\mathbf{r}$, mRRD 规模: (W, L, I_{max})

输出: 解码结果: $\hat{\mathbf{u}}$

$S, F \leftarrow \emptyset$

For $i \leftarrow 1$ **to** W **do**

  $\Sigma \leftarrow \text{RandomInt}() \bmod n$

  $\mathbf{w} \leftarrow \mathbf{r} >> \Sigma$

  For $j \leftarrow 1$ **to** L **do**

    $\mathbf{w} \leftarrow I_{max}$次 sum-product/min-sum($\mathbf{w}$)

    $\hat{\mathbf{u}} \leftarrow \text{HardDecision}(\mathbf{w})$

    If $\hat{\mathbf{u}}H^T = 0$ **then**

      $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{u}} << \Sigma$ //将比特重排至初始顺序

      $S \leftarrow S \cup \{\hat{\mathbf{u}}\}$ //将$\hat{\mathbf{u}}$添加到候选集

      Break

    Else

      $\mathbf{w} \leftarrow \mathbf{w} >> \sigma$

      $\Sigma \leftarrow (\Sigma + \sigma) \bmod n$ //记录右移总位数

    End if

  End for

$\Sigma \leftarrow (\Sigma - \sigma) \bmod n$

$\hat{\mathbf{u}} \leftarrow \hat{\mathbf{u}} << \Sigma$ //将比特重排至初始顺序

$F \leftarrow F \cup \{\hat{\mathbf{u}}\}$ //将$\hat{\mathbf{u}}$添加到缺省集

算法 3.4: 循环 mRRD

End for

If $|S| = 0$ **then**

$S \leftarrow F$ //没有合法码字，从缺省集中挑选结果

End if

$\hat{\mathbf{u}} \leftarrow \mathrm{LMS}(S)$

Return $\hat{\mathbf{u}}$

本文所提出的方法是使用随机循环移位来代替 mRRD 中每条流水线的第一个随机置换，以使得每条流水线有不同的起点。否则 mRRD 的多条流水线都会产生同样的输出，失去其意义。在这之后，使用算法 3.3 中得出的最佳置换代替原 mRRD 中的所有其他随机置换。由于所有的随机置换都被随机循环移位或者固定循环移位所取代，因此我们将新的修改后的 mRRD 称为神经网络循环 mRRD。其具体细节如算法 3.4 所示。其中函数RandomInt()是取随机整数、HardDecision()是硬判决（将小于 0 的 LLR 值转换为比特 1，大于零的 LLR 值转换为比特 0）。

3.4 实验

3.4.1 实验环境

本章节中所有的训练以及仿真都使用 BI-AWGN 信道，在迭代过程中所有的 LLR 都被截断到$[-20,+20]$以防止梯度爆炸。训练集大小为10^6，训练集中的向量均为被随机高斯噪声（信噪比为 3-5dB）污染的全零码字。在测试时则使用指定信噪比的随机高斯噪声污染的随机码字。本文中的超参数依经验设置如下：批大小为 200，学习率为 0.01，优化器为 RMSprop[43]，除学习率以外的优化参数使用 PyTorch 的默认值，其他环境配置如表 3.2 所示。

表 3.2. 实验软硬件环境

Table 3.2 The software and hardware environment of experiments.

名称	配置
操作系统	Ubuntu 18.04.5 LTS
Python	3.7
NumPy	1.19.1
PyTorch	1.7.0
CUDA	11.7

表 3.2. 实验软硬件环境（续）

Table 3.2 The software and hardware environment of experiments (continued).

名称	配置
中央处理器	Intel i9-10980XE
图像处理器	2* NVIDIA RTX 3090
内存	64 GB

3.4.2 评价指标

本文对于解码器的评价主要从两个角度出发：解码准确率、解码复杂度。对于解码准确率我们使用两个指标：比特错误率 (bit error rate, BER) 和帧错误率 (frame error rate, FER)，具体计算如公式(3.11)所示。BER 反映了比特数据流中出现错误比特的概率，而 FER 强调数据发送时每一帧的完整性，因为在实际中数据是按帧发送的，若一帧内出现错误，则整个帧的数据都需要重新传输或丢弃。

$$\begin{aligned}\mathrm{BER} &= \frac{\text{信息位中总错误比特数}}{\text{总信息位数}},\\ \mathrm{FER} &= \frac{\text{含错误}(\text{信息位和校验位})\text{帧数}}{\text{总帧数}}.\end{aligned} \tag{3.11}$$

对于解码复杂度，我们主要使用解码器的最大迭代次数、在各个 SNR 下的平均迭代次数以及每次迭代所需的运算次数进行分析。在信噪比很低时，接收向量中包含较多错误，此时解码所需的迭代次数常常达到最大值。平均迭代次数则反映了在不同噪声强度下分别需要的解码迭代次数。每次迭代所需的运算次数反映了不同算法在单次迭代中的操作类型和数量对比。

3.4.3 神经网络 Sum-product/Min-sum 和原算法的对比

为了探究神经网络算法的有效性，我们首先对比神经网络 sum-product 和 TAMS 算法及其原版算法之间的性能差异。如图 3.10 所示，图例中 NSP、NTAMS、SP、TAMS 分别对应神经网络 sum-product、神经网络 TAMS、sum-product、TAMS 的性能。可以看到神经网络 sum-product 和 TAMS 在错误率上的表现都好过其原版算法。其中神经网络 sum-product 算法在 BER 为2×10^{-4}时相比原版 sum-product 取得了 1.1 dB 的增益。神经网络 TAMS 算法在四者中表现最佳，其在 BER 为4×10^{-5}时相比原版 TAMS 取得了 0.6 dB 的增益，在 BER 为1×10^{-4}时和神经网络 sum-product 拥有类似的性能但在更高 SNR 下表现更优。除此之外，神经网络 TAMS 相比神经网络 sum-product 算法消除了求双曲正切和反双曲正切的步骤，运算量更小。因此在本章中若无特殊说明，使用的神经网络算法均基于神经网络 TAMS。

图 3.11 展示了 TAMS 和神经网络 TAMS 的错误分布对比，错误频数是在信噪比为 8 dB、总错误数超过 5 000 时统计的。可以看到神经网络 TAMS 算法对比 TAMS 算法的错误反而更加集中于陷阱集，这也验证了前文中的结论，即小型陷阱集是发生错误的主要因素。尽管图 3.10 显示神经网络 TAMS 比 TAMS 的准确率改进了很多（在 8 dB 的误码率约为 TAMS 的 1/10），但是在错误更少的情况下，神经网络 TAMS 的错误反而更加集中发生于小型陷阱集，这说明神经网络并不是通过缓解陷阱集效应来提高解码准确率。

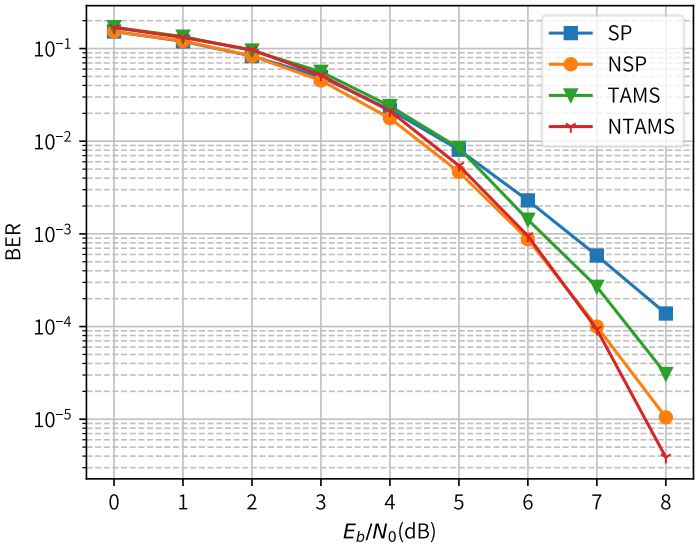

图 3.10 神经网络解码器与其原版算法的 BER 性能对比

Fig. 3.10 Performance comparison between different neural networks and their original versions.

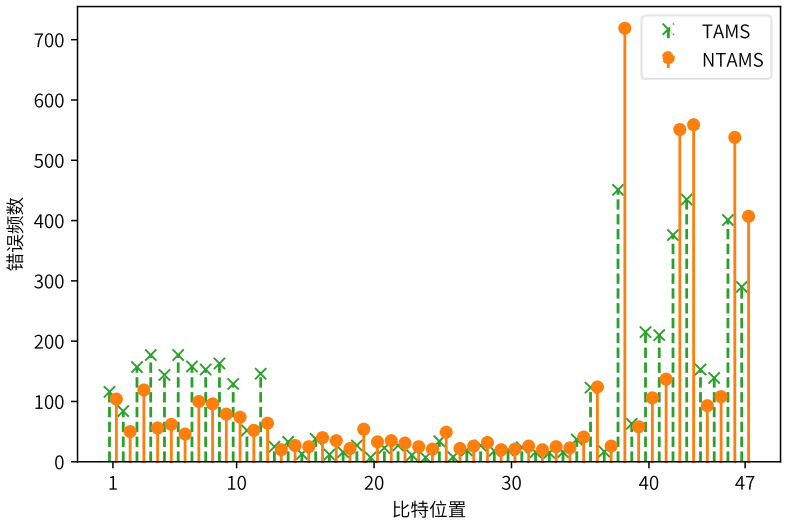

图 3.11 神经网络 TAMS 与原版 TAMS 的错误分布对比

Fig. 3.11 Comparison between neural TAMS and original TAMS on the error distribution.

3.4.4 神经网络 mRRD/循环 mRRD

本小节将对比基于神经网络的 mRRD 和循环 mRRD 的性能，以证明在给定规模的情况下所提出的神经网络循环 mRRD 能取得和神经网络 mRRD 相似的性能且算法更加简单。除此之外下文中的性能对比图中还加入了 DS 算法的性能作为基准，DS 算法是一类硬判决算法，可以保证能纠正$\lfloor d/2 \rfloor$内的错误。DS 算法是目前已知较快的 QR 码解码算法，但其在对长码解码时仍显得过于缓慢。这是因为 DS 算法每解码一个码长为n，信息位为k的码字就需要$(k-1)+(k-1)\sum_{i=1}^{\lfloor t/2 \rfloor} C_k^i$次实数加法以及$k+\sum_{i=1}^{\lfloor t/2 \rfloor} iC_k^i$次有限域加法。本小节中的 BER/FER 曲线（分别对应图中实线和虚线）都是在每个 SNR 上收集到 100 个错误帧后终止的。循环 mRRD 中的循环移位次数由算法 3.3 得到。下文中若无特殊说明，算法 3.3 的输入陷阱集均为满足$a+b<7, a<4$的基本陷阱集。图中的 NmRRD、NCmRRD、DS 分别代表神经网络 mRRD、神经网络循环 mRRD、DS 算法。图中所有的 mRRD 算法及其变种均使用 (5, 10, 5) 的规模。训练神经网络时使用的损失函数为式(3.8)对应的多损失 BCE。

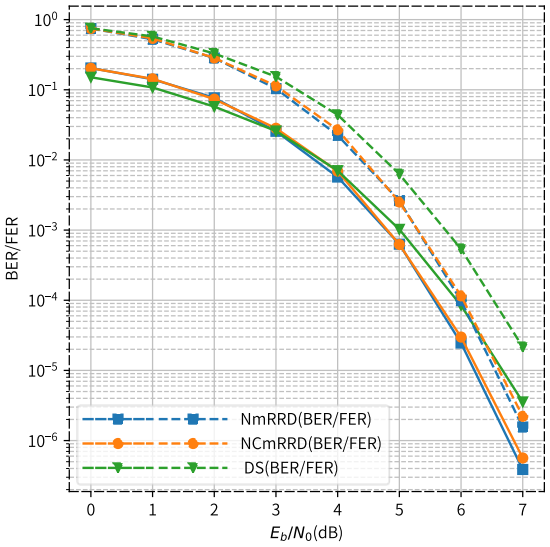

图 3.12 不同解码器在 QR (47, 24) 码上的错误率对比

Fig. 3.12 BER/FER performance of different decoders for the QR (47, 24).

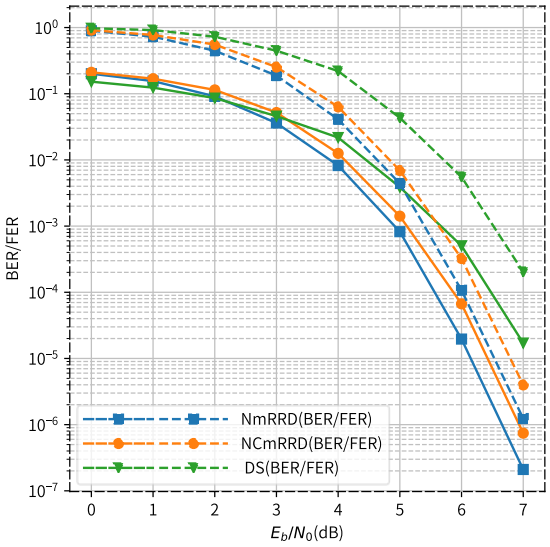

图 3.13 不同解码器在 QR (71, 36) 码上的错误率对比

Fig. 3.13 BER/FER performance of different decoders for the QR (71, 36).

图 3.12 对应不同解码器在 QR (47, 24) 码上的性能。相比 DS 算法，神经网络 mRRD 在 BER 为4×10^{-6}时取得了 0.5 dB 的增益。而神经网络循环 mRRD 相比神经网络 mRRD 只有轻微的性能损失，在 BER 为10^{-6}时只有 0.08 dB 的损失。相比

神经网络 mRRD 需要随机从整个置换群中选择置换，本文提出的神经网络循环 mRRD 只使用算法 3.3 提供的右移 22 位以及其他随机移位。除此之外，神经网络 mRRD 和循环 mRRD 在低信噪比下 BER 略高于 DS 算法但是在全 SNR 范围下 FER 都低于 DS 算法。这可能是因为 mRRD 算法在遇到无法解码的接收向量时输出的解码结果相比 DS 算法引入了更多的错误比特。

图 3.13 展示了不同解码器在 QR (71, 36) 码上的性能对比。其中神经网络 mRRD 对比 DS 算法在 BER 为2×10^{-5}时取得了 1.0 dB 的增益，而神经网络循环 mRRD 在 BER 为10^{-6}时相比神经网络 mRRD 有 0.3 dB 的性能损失。对于该码算法 3.3 输出的结果为循环右移 36 位。

图 3.14 展示了神经网络 mRRD 和神经网络循环 mRRD 在 QR (73, 37) 码上的 BER/FER 性能。与 DS 算法相比，神经 mRRD 在 BER 为3×10^{-6}时获得了约 0.5 dB 的增益。神经网络循环 mRRD 中选择的循环移位是右移 31 位，与神经网络 mRRD 相比，在 BER 为10^{-6}时，性能损失约为 0.1dB。

图 3.15 显示了神经网络 mRRD 和神经网络循环 mRRD 在 QR (97, 49) 码上的 BER/FER 性能。与 DS 算法相比，神经 mRRD 在误码率为2×10^{-6}时获得了约 0.4dB 的增益。神经网络循环 mRRD 选择的循环移位是 31 位右移。其与神经网络 mRRD 相比，在误码率为10^{-6}时，性能损失约为 0.04 dB。

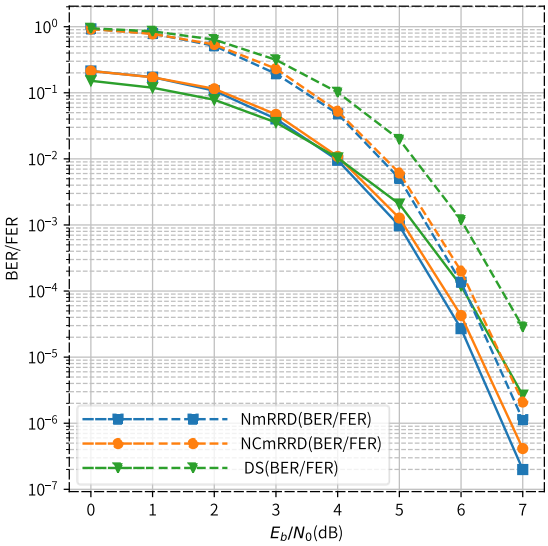

图 3.14 不同解码器在 QR (73, 37) 码上的错误率对比

Fig. 3.14 BER/FER performance of different decoders for the QR (73, 37).

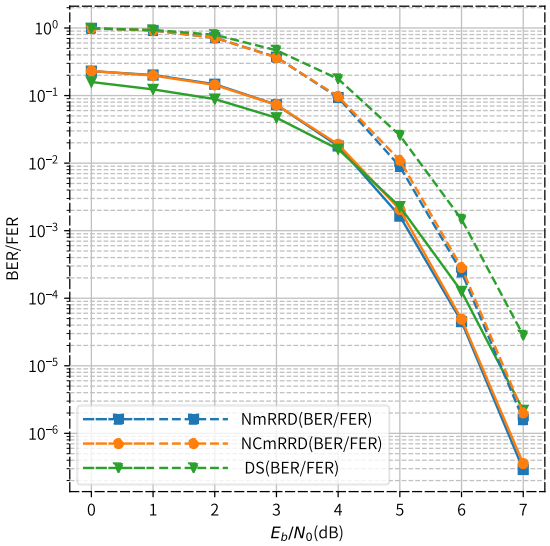

图 3.15 不同解码器在 QR (97, 49) 码上的错误率对比

Fig. 3.15 BER/FER performance of different decoders for the QR (97, 49).

3.4.5 在更长 QR 码上的实验

对于码长超过 200 的 QR 码，我们使用了不同的超参数集来训练神经网络解码器。训练集包含了2×10^6个全零码字，这些码字被相应的噪声（信噪比从 1 到 8 dB 不等）所污染。优化器的学习率设为 0.001，其他超参数保持不变。此外，我们使用普通的 BCE 损失函数而不是多损失 BCE。在下列的性能对比中不会出现 DS 算法的性能，因为运行 DS 算法需要已知码的最小汉明距离，而下列图中所示码长高于 200 的 QR 码的最小汉明距离仍然未知，此外 DS 算法的复杂度增长极快，在长度超过 127 时就很难得出结果，在长度超过 200 的 QR 码上更是几乎不可能得到可靠的性能数据。本小节中的迭代次数依然设为 5 次，mRRD 的规模设为 (5, 10, 5)。

图 3.16 展示了 TAMS 算法、神经网络 TAMS（对应图中 NTAMS）算法、神经网络循环 mRRD（对应图中 NCmRRD）算法在 QR (241, 121) 码上的 BER/FER 性能。其中神经循环 mRRD 使用了 191 位的循环右移（即 50 位循环左移）。输入陷阱集为$a < 4, a + b < 8$的基本陷阱集。图 3.17 显示了不同解码器在 QR (263, 132) 码上的 BER/FER 性能。采用了 232 位的循环右移（也就是 31 位的循环左移），输入陷阱集为$a < 5, a + b < 8$的基本陷阱集。可以看到当 BER 为1×10^{-4}时，神经网络循环 mRRD 在 QR (241, 121) 码上相比 TAMS 有接近 1.6 dB 的改善；在 BER 为3×10^{-6}时相比神经网络 TAMS 也有超过 0.9 dB 以上的增益。对于 QR (263, 132) 码而言，神经网络循环 mRRD 在 BER 为1×10^{-4}时相比 TAMS 有超过 1.5 dB 的增益；在 BER 为1×10^{-5}时相比神经网络 TAMS 有接近 1 dB 的增益。

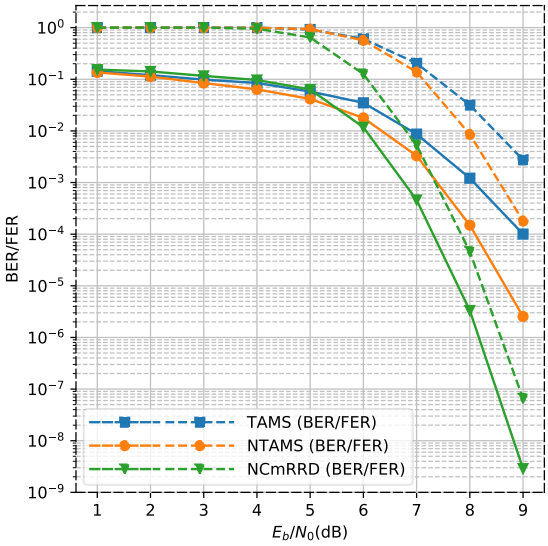

图 3.16 不同解码器在 QR (241, 121)的误码率/帧错误率对比

Fig. 3.16 BER/FER performance of different decoders for the QR (241, 121).

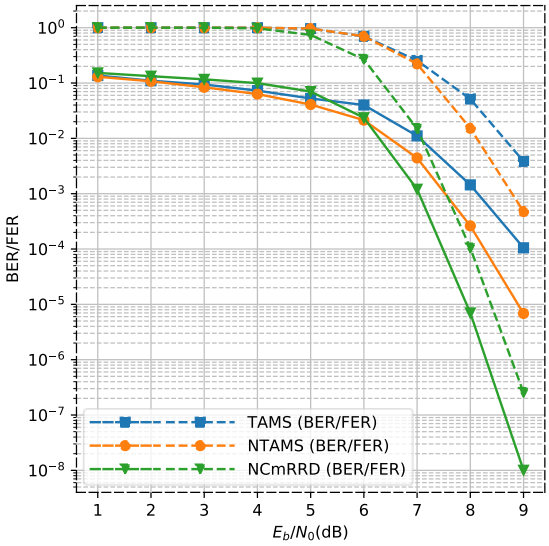

图 3.17 不同解码器在 QR (263, 132)的误码率/帧错误率对比

Fig. 3.17 BER/FER performance of different decoders for the QR (263, 132).

3.5 本章小结

在本章中，我们提出了几种基于神经网络的方法对二进制循环码进行解码。我们观察到，BP 类算法在解码密度较高的循环码时性能主要受小陷阱集的影响，而不是四环。即使是神经网络 BP 类算法也同样无法缓解小型陷阱集引发的性能下降。基于这一现象，我们所提出的神经网络循环 mRRD 通过使用循环移位将节点从陷阱集中移出。实现了与神经网络 mRRD 相似的性能，同时降低了实现复杂度和内存占用。神经网络 mRRD 和神经网络循环 mRRD 在误码率性能上都优于 DS 算法，且渐近时间复杂度更低。接着，我们使用神经网络循环 mRRD 获得了两个较长 QR 码的 BER/FER 性能。对于这二者，在不知道它们的最小汉明距离的情况下，使用传统的 DS 算法无法对其进行解码，而提出的基于 mRRD 的算法则不受此约束。

4 逼近循环码的最大似然解码性能

第 3 章介绍了多种神经网络解码器，其中神经网络循环 mRRD 算法获得了比 DS 算法更好的性能，但两者离 ML 解码性能均有很大差距。在本章，我们将尝试进一步在多种码字上逼近 ML 性能。我们将首先论述如何快速求解中短长度码字的 ML 性能，并通过引入 Chase 算法尝试改进神经网络循环 mRRD 的性能。此外，本章还尝试基于分层 min-sum 算法，对神经网络解码器的网络结构进行优化，在提升解码准确率的同时降低解码的时间复杂度、空间复杂度以及训练耗时。在本章中我们将会使用不同长度和类型的码进行测试，并与文献[33]1771-1780 中提出的神经网络解码器包括循环不变 BP (cyclically equivariant BP, CEBP)解码器，表解码器 (list decoder, LD) 以及 ML 解码器进行对比。

4.1 基于线性规划以及整数规划的 ML 解码器

线性分组码的 ML 解码性能可以用线性规划和整数规划来求得。众所周知，给定接收向量$\mathbf{r}$并设其转置为$\mathbf{r}^T$，则任何二进制线性分组码的 ML 解码都可以被表述为一个优化问题，即搜索满足以下条件的码字$\mathbf{u}$：

$$\min \mathbf{r}^T\mathbf{u} \quad \text{s.t.}\, \mathbf{u} \in \mathcal{C}. \tag{4.1}$$

其中$\mathcal{C}$是所有合法码字构成的集合。由于$\mathcal{C}$是二进制码字构成的集合，所以这个问题是一个整数规划问题。为了近似 ML 解码过程以快速得到结果，Feldman 等人[44]松弛了可行域，将其从$\mathcal{C}$改为$\mathcal{C}$的凸包，即所谓的码字多边形。码字多边形中既包含了原来的码字，也包含了一些非整数向量。松弛后的问题可以通过线性规划 (linear programming, LP) 来求解。如果 LP 解码器得到一个整数解，那么该解就必然是原问题的最优解，也就是一个 ML 码字。在文献[45]中，Zhang 和 Siegel 提出了一种高效的基于线性规划的解码器，称为 ACG-ALP 解码器。该解码器进一步加速了 LP 解码的过程。本文基于 ACG-ALP 解码器来获得码的 ML 性能，具体如下：

（1）首先使用 ACG-ALP 算法解码接收向量，若找到的解是整数解，那么停止算法并输出这个解。

（2）若 1）未找到整数解，使用其输出的非整数解作为起点构造整数规划并使用 ACG-ALP 算法中的所有约束进行剪枝并求解。

上述方法是基于这样的想法：ACG-ALP 算法给出的结果即使不是 ML 码字也应该接近于 ML 码字。因此，在执行 ACG-ALP 解码后使用整数规划应该比单独使用整数规划来寻找 ML 码字要快得多。

4.2 基于 Chase-II 算法逼近循环码的 ML 性能

4.2.1 Chase-II-mRRD 算法

著名的 Chase-II 算法[46]是一类软判决算法，其原理是通过翻转接收向量中不可靠的比特可以大概率减少接收向量中的错误数量，从而提高解码的准确率。在 Chase-II 算法中，给定接收向量$\mathbf{r}$，算法会首先生成一个测试向量集合$\mathbb{T}$。取$\mathbf{r}$中$\lfloor d/2 \rfloor$个最不可靠的位置（即绝对值最小的位置，在概率上而言这些位置对应的噪声值偏大），在这$\lfloor d/2 \rfloor$个位置中穷尽所有的 0/1 排列而其他$n-\lfloor d/2 \rfloor$个位置均为0。$\mathbb{T}$就是所有这样的向量构成的集合，所以有$|\mathbb{T}| = C_{\lfloor d/2 \rfloor}^{0} + C_{\lfloor d/2 \rfloor}^{1} + C_{\lfloor d/2 \rfloor}^{2} + \cdots + C_{\lfloor d/2 \rfloor}^{\lfloor d/2 \rfloor} = 2^{\lfloor d/2 \rfloor}$。其中$\lfloor \cdot \rfloor$代表向下取整，$d$代表码的最小汉明距离。在生成测试向量集合$\mathbb{T}$后，Chase-II 算法对$\mathbf{r}$进行硬判决得到$\mathbf{x}$，即对$i \in \{1,2,\ldots,n\}$，若$r_i > 0$则$x_i = 0$，若$r_i \le 0$则$x_i = 1$。最后对所有$\mathbf{t} \in \mathbb{T}$，Chase-II 算法都使用一个硬判决解码器对$\mathbf{t} \oplus \mathbf{x}$进行解码，其中$\oplus$是异或，然后从所有的解码结果中挑选一个最好的作为输出。

基于 Chase-II 算法和神经网络循环 mRRD 算法我们提出了 Chase-II-mRRD 算法。与原版 Chase-II 算法不同的地方在于$\mathbb{T}$中的向量从 0/1 向量变为$+1/-1$向量，而 Chase-II 算法中使用硬判决算法对$\mathbf{t} \oplus \mathbf{x}$进行解码这一步骤被替换为使用前文提出的神经网络循环 mRRD 算法对$\mathbf{t} \odot \mathbf{r}$进行解码，其中$\odot$为向量对应位置元素相乘。考虑到$\mathbf{t}$是$+1/-1$向量，这样做的效果就是对$\mathbf{r}$中某些元素的符号进行翻转。若恰好翻转到了符号错误的比特，则有可能提升后端神经网络循环 mRRD 算法的解码准确率。Chase-II-mRRD 算法不断对$\mathbf{t} \odot \mathbf{r}$进行解码直到满足文献[47]中提到的快速终止条件或$\mathbb{T}$中的所有测试向量都被使用过。这里的快速终止条件可以识别当前的解码结果是否是给定输入的 ML 码字。若当前的结果被认为是 ML 码字则可以保证无论$\mathbb{T}$中还有多少向量未被测试，其解码结果都不会好于当前结果。快速终止条件的具体表述如下，令：

$$z_i = \begin{cases} 0, \text{若} r_i \ge 0 \\ 1, \text{若} r_i < 0 \end{cases} \tag{4.2}$$

$$D_0(\mathbf{v}) = \{i | v_i = z_i\}, D_1(\mathbf{v}) = \{i | v_i \ne z_i\}. \tag{4.3}$$

定义相关差：

$$\lambda(\mathbf{r}, \mathbf{v}) = \sum_{i \in D_1(\mathbf{v})} |r_i| . \tag{4.4}$$

假设$D_1(\mathbf{v})$中有$n_{\mathbf{v}}$个元素，那么$D_0(\mathbf{v})$中就有$n - n_{\mathbf{v}}$个元素，按照可靠性将其进行排序。若排序后的结果为$D_0(\mathbf{v}) = \{l_1, l_2, \ldots, l_{n-n_{\mathbf{v}}}\}$，那么对于任意$i < j$有$|r_{l_i}| < |r_{l_j}|$。此外定义该集合中前$j$个元素为:

$$D_0^{(j)}(\mathbf{v}) = \{l_1, l_2, \ldots, l_j\}. \tag{4.5}$$

定义$\delta = d - n_{\mathbf{v}}$，$G_T(\mathbf{v}, d) = \sum_{i \in D_0^{(\delta)}(\mathbf{v})} |r_i|$，若：

$$\lambda(\mathbf{r},\mathbf{v}) \le G_T(\mathbf{v},d), \tag{4.6}$$

那么解码结果**v**就是接受向量**r**对应的 ML 码字。公式(4.6)是一个充分条件，若该式被满足则解码结果必然是 ML 码字；反之，不满足公式(4.6)的码字未必一定不是 ML 码字。Chase-II-mRRD 算法的流程图如图 4.1 所示。

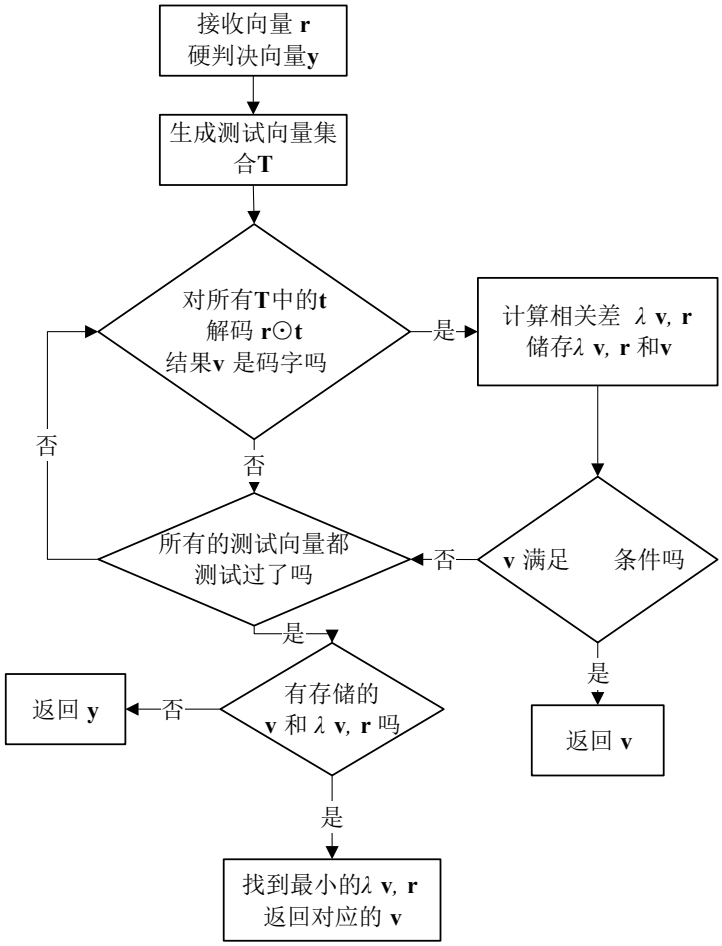

图 4.1 Chase-II-mRRD 算法流程图

Fig. 4.1 The flow chart of Chase-II-mRRD algorithm.

4.2.2 仿真性能及分析

本小节主要检验 Chase-II 算法和提出的 Chase-II-mRRD 算法之间的性能差异。其中原版 Chase-II 算法的内核为 DS 算法。ML 性能则是本章所提出的基于线性规划以及整数规划的解码器得到的。和第 3 章相同，本小节中的神经网络循环 mRRD 规模也使用 (5, 10, 5)。本小节中的 FER 结果在 0 到 5 dB 时是收集 100 个错误帧得到的。而在 5.5 以及 6 dB 时由于错误率已经很低，想要收集到 100 个错误帧的耗时过长，所以在这两种情况下的 FER 是收集到 20 个错误帧时计算得到的。

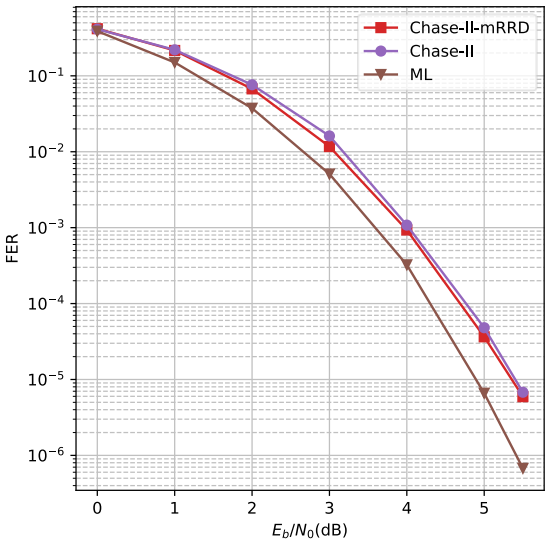

图 4.2 不同解码器在 QR (47, 24) 码上的帧错误率对比

Fig. 4.2 FER performance of different decoders for the QR (47, 24).

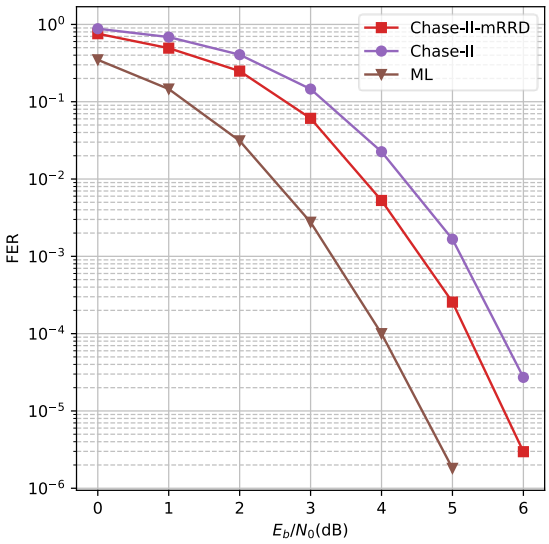

图 4.3 不同解码器在 QR (71, 37) 码上的帧错误率对比

Fig. 4.3 FER performance of different decoders for the QR (71, 37).

如图 4.2 所示，与原 Chase-II 算法相比，Chase-II-mRRD 在 QR (47, 24) 码上的改进非常小。造成这一现象的原因可能是因为两者的性能都已经比较接近 ML 性能。当 FER 为7×10^{-6}时，Chase-II-mRRD 的性能与 ML 性能相差 0.5dB。图 4.3 显示了不同解码器在 QR (71，36) 码上的 FER 结果。与原始的 Chase-II 算法相比，当 FER 为3×10^{-5}时 Chase-II-mRRD 的增益为 0.5 dB。在 FER 为3×10^{-6}时，Chase-II-mRRD 和 ML 性能之间的差距是 1.2 dB。

图 4.4 显示了不同解码器在 QR (73, 37) 码上的 FER 性能比较。与原始的 Chase-II 算法相比，Chase-II-mRRD 在 FER=3×10^{-6}时获得约 0.4 dB 的改进；此时，Chase-II-mRRD 和 ML 之间的差距是 1.1 dB。图 4.5 显示了不同解码器在 QR (97, 49)码上的错误性能比较。与原始的 Chase-II 算法相比，Chase-II-mRRD 在 FER=4×10^{-6}时获得约 0.4 dB 的改进，此时，Chase-II-mRRD 和 ML 之间的差距是 1.4 dB。

我们注意到尽管随着码长的增长，Chase-II-mRRD 算法和 ML 的 FER 性能差距逐渐拉大，但是我们提出的 Chase-II-mRRD 算法始终在 FER 性能上优于 Chase-II 算法。

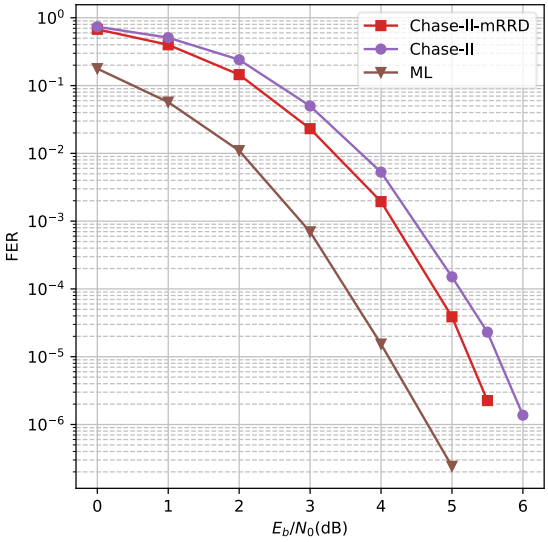

图 4.4 不同解码器在 QR (73, 37)的帧错误率对比

Fig. 4.4 FER performance of different decoders for the QR (73, 37).

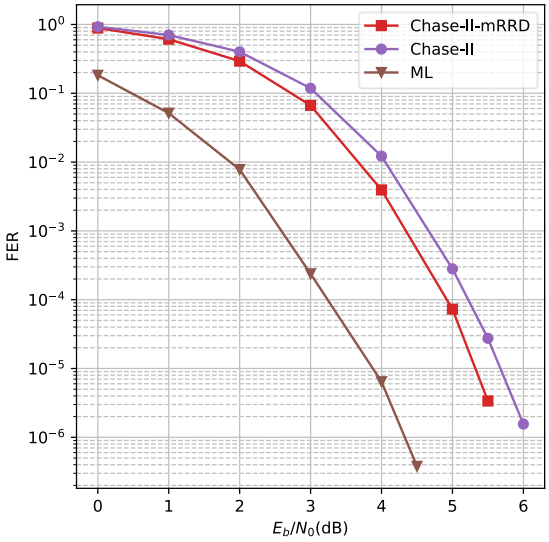

图 4.5 不同解码器在 QR (97, 49)的帧错误率对比

Fig. 4.5 FER performance of different decoders for the QR (97, 49).

4.2.3 复杂度分析

本节所提出的 Chase-II-mRRD 和第 3 章所提出的 NCmRRD 方法尽管在解码时进行了多次 min-sum 迭代，并在每次迭代时引入了乘法权重。但这些操作大多数可以并行运行。假设 Tanner 图中的边数为e，每次神经网络 min-sum 迭代平均需要比原始的 min-sum 迭代多$2e$次乘法。但延迟几乎保持不变，因为不同的节点可以同时处理传入的信息。对于神经网络循环 mRRD(W, L, I_{max})，在最坏的情况下需要$W \times L \times I_{max}$次 min-sum 迭代，而在最好的情况下只需要$W$次迭代，因为在最好的情况下每条流水线只需要执行 1 次迭代。考虑到不同流水线之间没有数据依赖性，W条流水线可以同时解码不同的输入。因此在并行解码时最坏情况下的延迟减少到$L \times I_{max}$次迭代，而最佳情况下的迭代次数为I_{max}次。对于我们的 NCmRRD (5, 10, 5)，并行解码的延迟在最坏情况下是 50 次迭代，在最好情况下是 5 次迭代。Chase-II-mRRD 在最坏情况下的复杂度是$2^{\lfloor \frac{d}{2} \rfloor} \times W \times L \times I_{max}$次迭代，其中$d$是码的最小汉明距离。和 mRRD 类似，Chase-II-mRRD 的运算大部分也可以并行化。若我们依次对测试向量集的每个向量进行解码并将 mRRD 的部分并行化，则此时的复杂度最高为$2^{\lfloor \frac{d}{2} \rfloor} \times L \times I_{max}$。

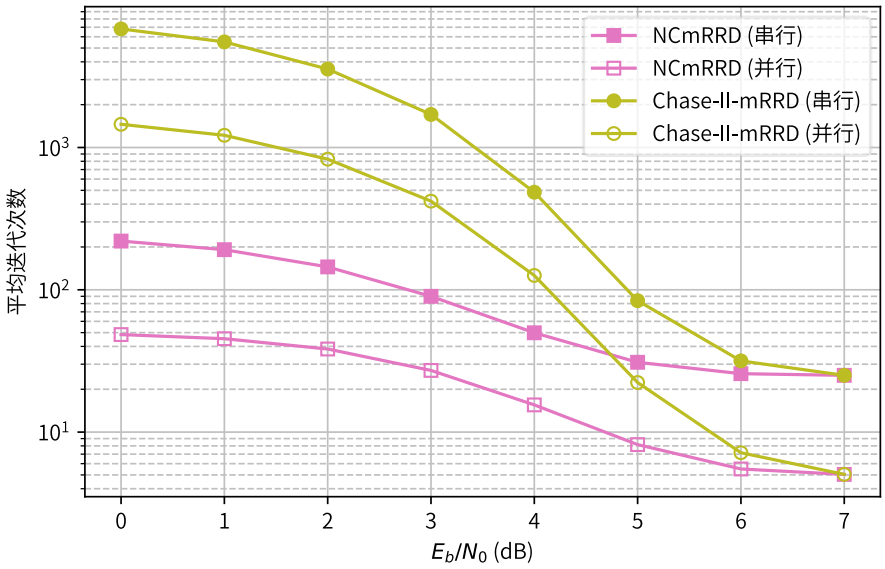

图 4.6 不同解码器在 QR (47, 24)的平均复杂度对比

Fig. 4.6 Average complexity on the QR (47, 24).

想要分析上述解码器的平均复杂度通常很困难。因此，我们通过在不同信噪比条件下分别进行仿真来获得平均的时间复杂度。在每个信噪比下我们对 10 000 个随机帧进行解码并统计每一帧解码得到输出结果或解码失败时所需的迭代次数。统计结果如图 4.6 所示，图例中的“串行”是指将所有流水线的迭代次数相加，而“并行”表示所有流水线中最晚输出结果的那条流水线所需的迭代次数。图中串行或并行迭代复杂度的计算方法是将每个接收向量对应的迭代次数相加，然后除以接收向量总数。可以看到，在 SNR 为 6 dB 时，神经网络循环 mRRD 的平均复杂度接近于我们之前推导的最佳复杂度（串行约 25 次迭代，并行约 5 次迭代），在 SNR 为 0 dB 时，复杂度最高（串行约 220 次迭代，并行 48 次）。在低信噪比区域中，Chase-II-mRRD 的复杂性比神经网络循环 mRRD 高得多，例如在信噪比为 0 dB 时，Chase-II-mRRD 大约慢 31 倍，但在信噪比为 7 dB 时，Chase-II-mRRD 与神经网络循环 mRRD 几乎一样快。

4.3 基于分层 min-sum 及 mRRD 算法逼近循环码的 ML 性能

4.3.1 权重分配以及分层 min-sum 算法

在第 3 章，我们发现尽管神经网络 TAMS 算法能大幅度降低错误率，但是并非通过缓解四环效应和陷阱集效应这两种途径。因此我们认为神经网络主要通过收缩消息传递的幅度来提升解码准确率。考虑到为每条边分配不同权重的初衷是通过对环路上的边分配不同权重从而缓解短环的效应，而短环并非影响性能的最

大因素。因此为每次迭代分配权重和逐边分配权重相比，在准确率性能上可能相差无几，但在权重数目以及训练速度上则相差甚远。例如对于 BCH (63, 45) 码，其 Tanner 图的边数$n_e = 432$，行重$n_r = 24$。若迭代次数设为I_{max}，则神经网络 BP 类算法对应的神经网络权重数量为$2 \times n_e \times I_{max} = 864I_{max}$。CEBP 算法的权重情况则比较特殊。CEBP 使用了一种循环冗余校验矩阵，即将原校验矩阵(2.10)扩展为：

$$\mathbf{H}_{cyc} = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 \\ 0 & 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ h_{k-3} & \cdots & h_0 & 0 & \cdots & h_k & h_{k-1} & h_{k-2} \\ h_{k-2} & h_{k-1} & \cdots & h_0 & 0 & \cdots & h_k & h_{k-1} \\ h_{k-1} & h_{k-2} & h_{k-3} & \cdots & h_0 & 0 & \cdots & h_k \end{bmatrix}, \tag{4.7}$$

扩展后的矩阵大小从$(n-k) \times n$变为$n \times n$。CEBP 除了使用了特殊的矩阵外，还对权重做了特殊的共享分配，其对应的权重数量为$2 \times n_r \times n_r \times I_{max} = 1152I_{max}$。而普通的神经网络 BP 类算法在使用$\mathbf{H}_{cyc}$作为校验矩阵时的权重数量则为$2 \times n \times n_r \times I_{max} = 3024I_{max}$。倘若使用我们上文提出的权重分配方法，对每次迭代只分配一个权重，则无论使用何种矩阵，神经网络的总权重数目均为I_{max}。假定I_{max}取 10，各解码器在解码 BCH (63, 45) 码时所需的权重数量如表 4.1 所示。需要注意的是，CEBP 并不适用于$\mathbf{H}$且只能在$\mathbf{H}_{cyc}$上运行。但是当逐边分配权重的神经网络 BP 类算法也在$\mathbf{H}_{cyc}$上运行时，CEBP 的共享权重策略将使得其权重数比普通的神经网络 BP 类算法权重数更少。

表 4.1 不同算法在不同矩阵上解码 BCH (63, 45)码的权重数对比

Table 4.1 The number of weights for different decoders to decode BCH (63, 45) when using different matrices.

算法	权重数($\mathbf{H}$)	权重数($\mathbf{H}_{cyc}$)
NTAMS	8 640	30 240
NLMS	10	10
CEBP	不适用	11 520

除了在权重分配方式上进行了改进，我们还提出使用分层算法构建神经网络。前文提到的 BP 类算法的更新方式为泛洪式，即在一次迭代中所有的变量节点和校

验节点同时发送信息并更新。在分层式算法中，变量节点根据其与校验节点的连接关系被分成若干组，每组节点依次进行更新。这样做的好处是在同一次迭代中后更新的节点可以利用之前更新的节点更新后的消息，从而加快收敛速度。文献[48]中作者证明了一般情况下异步更新的消息传递算法（在这里的情况中是分层 min-sum 算法）会比同步更新的算法（在这里是泛洪传播的 min-sum 算法）更快收敛到不动点（即合法码字）。分层神经网络 min-sum 的初始化以及输出步骤和前文介绍的 min-sum 相同，但是迭代步骤变化如下：

依次对每一个校验节点c_j及其相邻变量$v_i \in N(c_j)$进行如下操作：

（1）变量节点到校验节点更新，其中$l_{v_i}^1 = l_{v_i}^{ch}, l_{c_j \to v_i}^0 = 0$：

$$l_{v_i}^k = l_{v_i}^k - l_{c_j \to v_i}^{k-1} \tag{4.8}$$

（2）校验节点到变量节点更新：

$$l_{c_j \to v_i}^k = w^k \left(\prod_{v_{i'} \in N(c_j) \backslash \{v_i\}} \operatorname{sign}\left(l_{v_{i'}}^k\right) \right) \cdot \min_{v_{i'} \in N(c_j) \backslash \{v_i\}} \left| l_{v_{i'}}^k \right| \tag{4.9}$$

（3）变量节点更新：

$$l_{v_i}^{k+1} = l_{v_i}^k + l_{c_j \to v_i}^k \tag{4.10}$$

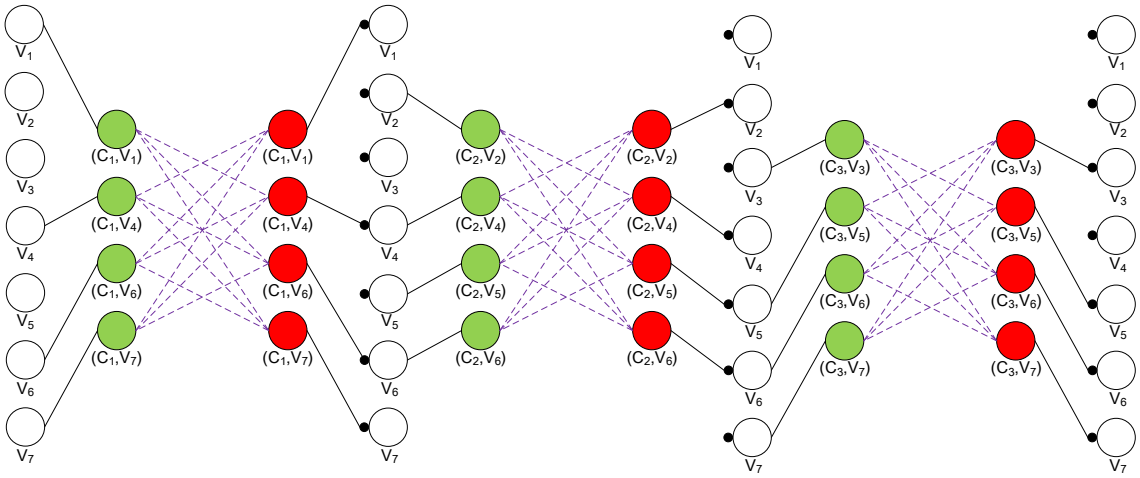

图 4.7 (7, 4)汉明码的神经网络分层 min-sum 算法结构图

Fig. 4.7 The structure of neural layered min-sum for (7, 4) Hamming code

新提出的神经网络分层 min-sum 解码器的网络结构如图 4.7 所示。其中紫色虚线代表可训练的权重（在同一层内权重共享），黑色小圆圈则代表加上前一层的输出 LLR，可以看到相比泛洪式 BP 算法对应的神经网络，分层式算法的层数更多但是每一层的规模更小，且本章提出的算法拥有更少的权重。

4.3.2 改进的 mRRD 算法

使用分层 min-sum 算法的好处是加快了解码器的收敛速度，但是并不能解决小型陷阱集所带来的问题。而本章第二小节所提出的 Chase-II-mRRD 可以在很短的循环码上获得较为接近 ML 解码的性能，但在其他长度的码字上离 ML 性能差距较大。其原因可能是 Chase-II 算法主要依据比特的可靠度进行翻转操作，而影响神经网络循环码解码器的主要因素是小型陷阱集，但是陷阱集的分布和比特可靠度的分布相互独立。为了进一步改进本章所提出的神经网络分层 min-sum 算法的纠错性能，我们改进了 mRRD 算法来消除陷阱集的影响。除此之外我们将前文提到的用于 Chase-II 算法的快速终止条件直接引入 mRRD 而不使用 Chase-II 算法以降低复杂度，这样做的好处是可以更加灵活地设置流水线数量从而在解码准确率和运行速度上找到更好的折中。这样做还节省了 Chase-II 算法所需的可靠度排序、比特翻转等步骤，也节省了用于存储测试向量集合的空间。

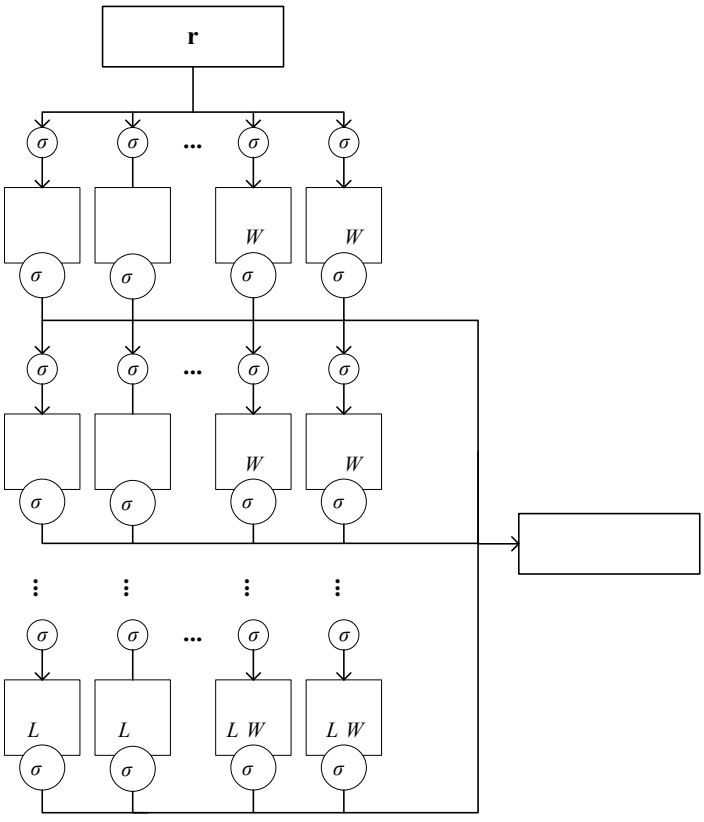

图 4.8 改进后的 mRRD 结构示意图

Fig. 4.8 The structure of improved mRRD.

改进的 mRRD 解码器的框架如图 4.8 所示，其中σ代表从置换群中抽取的随机置换，而σ^{-1}代表其逆置换。图中的每一个正方形“BP”代表一个迭代次数为I_{max}

的神经网络 BP 类解码器，而“ML”代表式(4.6)中的 ML 码字判别准则。每完成一轮置换、解码、逆置换后，成功译码的结果会被输入 ML 判决准则，若判定为为 ML 码字则可以提前退出；如果L轮后依然没有一个码字被认为时 ML 码字，则输出相关差（使用公式(4.4)计算）最小的码字；若所有解码器都没有输出码字，则宣布解码失败。这里我们并非只使用循环移位来进行 mRRD 的置换操作，这是因为想要逼近 ML 解码性能需要较高的流水线条数。但是循环移位的种类十分受限，当流水线条数增长时只使用循环移位无法提供足够的多样性，会导致多条流水线产生同样的结果，使得 mRRD 的性能退化。如果更高的流水线条数可以带来逼近 ML 解码的性能，以及相比 Chase-II-mRRD 算法更低的迭代次数，那么在置换步骤增加部分复杂度是可以接受的。改进后的神经网络 mRRD 具体如算法 4.1 所示。

算法 4.1: 改进 mRRD

输入:接收向量: $\mathbf{r}$, mRRD 规模: (W, L, I_{max})

输出: 解码结果: $\hat{\mathbf{u}}$

$S \leftarrow \emptyset, \mathbf{o} = \mathbf{r}$

For $i \in \{1, \dots, W\}$ **do**

  For $j \in \{1, \dots, L\}$ **do**

    随机从Per$(\mathcal{C})$中抽取置换π

    $\mathbf{o} \leftarrow I_{max}$次 min-sum$(\pi(\mathbf{o}))$

    $\hat{\mathbf{u}} \leftarrow$ HardDecision$(\mathbf{o})$

    $\mathbf{o} \leftarrow \pi^{-1}(\mathbf{o}), \hat{\mathbf{u}} \leftarrow \pi^{-1}(\hat{\mathbf{u}})$,

    If $\hat{\mathbf{u}}H^T = 0$ **then**

      $S \leftarrow S \cup \{\hat{\mathbf{u}}\}$ //将$\hat{\mathbf{u}}$添加到候选集

      If $\hat{\mathbf{u}}$是$\mathbf{r}$的 ML 码字

        Return $\hat{\mathbf{u}}$

      End if

      Break

    End if

  End for

End for

If $|S| \neq 0$ **then**

  Return $\text{argmin}_{\hat{\mathbf{u}} \in S} \lambda(\mathbf{r}, \hat{\mathbf{u}})$ //返回最有可能的合法码字

Else

  宣布解码失败

End If

4.4 实验

4.4.1 实验设置

由于提出的神经网络大幅度减少了网络权重的数量以及每一层的宽度，在训练时便可以使用更大的批大小。本章节提出的神经网络分层 min-sum（以下简称 NLMS）训练时使用的批大小为 2 000，学习率为 0.01，损失函数均使用普通的 BCE 而非多损失 BCE。NLMS、分层 min-sum (LMS)、CEBP 的解码器最大迭代次数都设置为 5。训练集使用随机噪声以及全零码字而测试集使用随机噪声和随机码字。

4.4.2 NLMS 的解码性能

图 4.9 显示，与使用逐边分配权重的解码器相比，无论是在 QR (47, 24) 码还是在 BCH (63, 45) 码上，我们提出的逐迭代分配权重的解码器在权重数量更少的情况下都实现了类似的性能。逐层分配的权重对于迭代中所有边上通过的信息都进行同样幅度的衰减，并不会因为信息从环上通过就进行不同的处理。而此时却可以达到和逐边分配权重同样的的性能。这也从另一个角度验证了前文对于神经网络无法针对四环和陷阱集进行缓解的猜想。

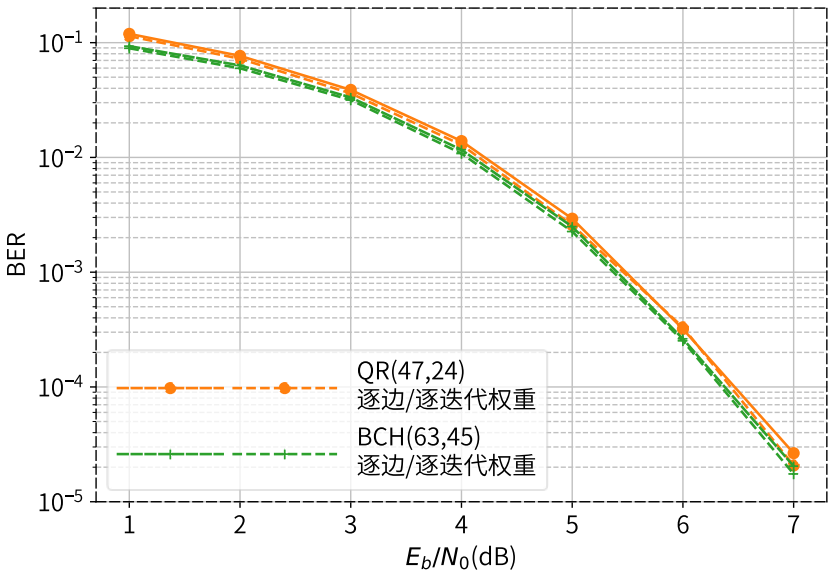

图 4.9 使用不同权重分配方式的 NLMS 解码 QR (47, 24)的 BER 性能对比

Fig. 4.9 BER performance of NLMS using different weight assignments on the QR (47, 24).

BCH (63, 45) 码的仿真结果见图 4.10。可以看出，LMS 在两个矩阵（即$\mathbf{H}$和$\mathbf{H}_{cyc}$）之间的误码率（BER）差异很小。然而，在$\mathbf{H}$和$\mathbf{H}_{cyc}$两种矩阵上，NLMS 解码器对比 LMS 解码器都取得了明显的增益，在 BER 为10^{-4}时，其改进幅度分别为 0.35 dB 和 1.12 dB。此外，在 BER 为4×10^{-6}时，NLMS 解码器对比 CEBP 解码器的性能

增益为 0.06dB。此外，我们发现使用$\mathbf{H}_{cyc}$的 NLMS 解码器明显优于使用原始矩阵$\mathbf{H}$的解码器，这种现象在其他码的性能对比中也可以看到。这也符合前文提到的四环并非是影响神经网络 BP 解码准确率的主导因素这一观点。因为$\mathbf{H}_{cyc}$的前$n-k$行与$\mathbf{H}$相同，且后k行由前面移位得到，故$\mathbf{H}_{cyc}$中的四环数量远超$\mathbf{H}$矩阵。解码器在$\mathbf{H}_{cyc}$的四环数远高过$\mathbf{H}$矩阵的情况下表现出了更低的误码率是因为尽管$\mathbf{H}_{cyc}$的四环数变多了，但其包含的陷阱集所对应的参数a,b也变大了。对于 QR (47, 24) 码而言，其$\mathbf{H}_{cyc}$矩阵中不包含$b<12$的陷阱集。而更大的陷阱集更不容易妨碍解码过程的收敛。

图 4.11 中显示了不同解码器在 BCH (63, 36) 码上的性能比较。我们注意到，在 BER 为2×10^{-4}的情况下，NLMS 解码器相比 LMS 解码器取得了 0.58 dB 的增益，而在使用$\mathbf{H}_{cyc}$的情况下则为 1.20 dB。此外，我们发现在 BER 为10^{-5}的情况下，NLMS 解码器比 CEBP 解码器有 0.37 dB 的增益。

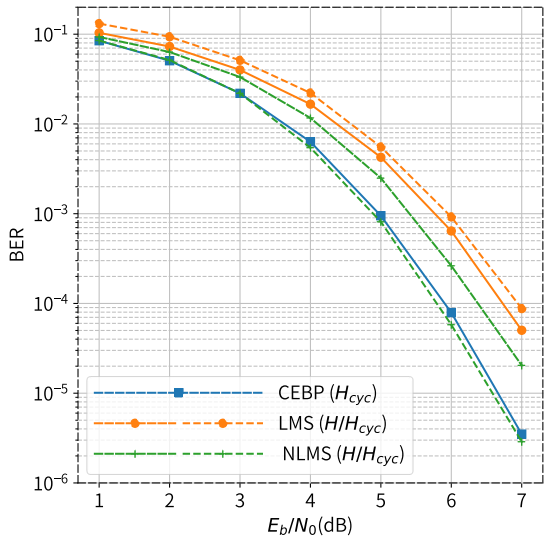

图 4.10 不同算法在不同矩阵上解码 BCH(63,45) 码的 BER 性能

Fig. 4.10 BER performance of different decoders on the BCH (63, 45) over different parity-check matrices.

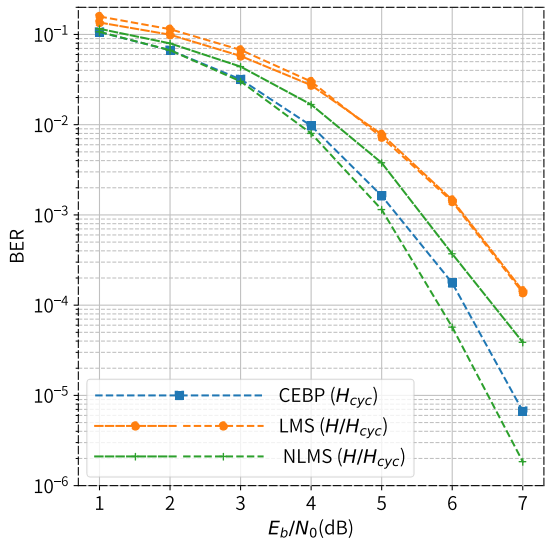

图 4.11 不同算法在不同矩阵上解码 BCH(63,36) 码的 BER 性能

Fig. 4.11 BER performance of different decoders on the BCH (63, 36) over different parity-check matrices.

不同解码器对 QR (47, 24) 码的误码率性能对比见图 4.12。可以看出，NLMS 解码器在矩阵$\mathbf{H}$上相比 LMS 解码器获得了 0.64 dB 的增益，而在 BER 为2×10^{-4}的情况下，NLMS 在$\mathbf{H}_{cyc}$上相比 LMS 的增益为 1.18 dB。我们还观察到，在 BER 为10^{-5}的情况下，NLMS 比 CEBP 解码器有 0.32 dB 的增益。

图 4.13 中描述了不同解码器在 QR (71, 36) 码上的性能曲线。在 BER 为 4×10^{-4}时，NLMS 解码器相比 LMS 解码器实现了 0.90 dB 的增益。NLMS 在使用$\mathbf{H}_{cyc}$矩阵进行解码的情况下相比 LMS 实现了 1.38 dB 的增益。此外，我们注意到，在 BER 为10^{-5}时，NLMS 解码器优于 CEBP 解码器 0.29 dB。

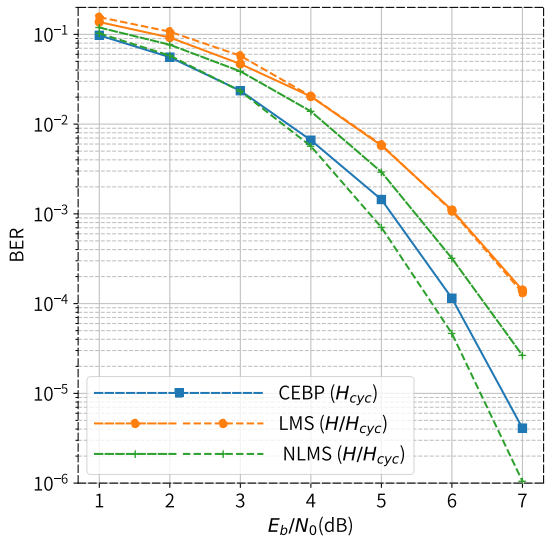

图 4.12 不同算法在不同矩阵上解码 QR (47, 24) 码的 BER 性能

Fig. 4.12 BER performance of different decoders on the QR (47, 24) over different parity-check matrices.

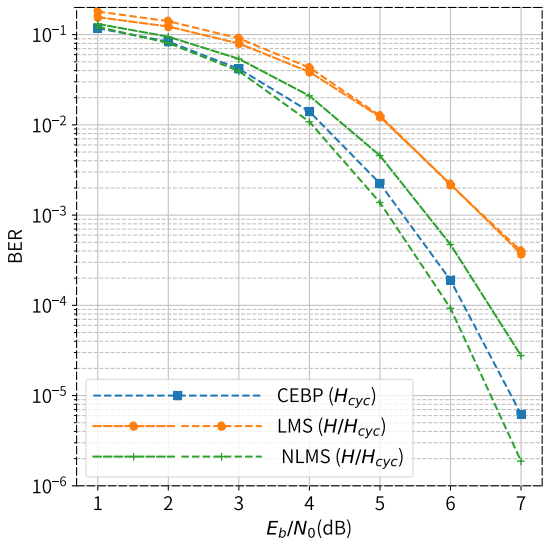

图 4.13 不同算法在不同矩阵上解码 QR (71, 36) 码的 BER 性能

Fig. 4.13 BER performance of different decoders on the QR (71, 36) over different parity-check matrices.

4.4.3 改进的 mRRD 的性能评估

尽管提出的 NLMS 算法在降低复杂度的情况大幅度提升了误码率性能，但是仍然和 ML 性能有较大差距。本章使用前文提到的改进的神经网络 mRRD（下文称 FNmRRD）算法尝试继续逼近 ML 性能。同时我们使用目前性能最好的表解码[33]1771-1780（下文称 LD）算法作为对比，以证明所提出算法的优越性。

从图 4.14 中可以看出，随着解码器宽度（流水线数量）W的增长，FNmRRD 和 LD 都可以达到接近 ML 的性能。在 FNmRRD 和 LD 使用图中性能最好的参数时，LD 在最坏情况下的复杂度是 FNmRRD 算法的 5 倍（$64\times20\times5=5\times(256\times5\times1)$）。此外，我们提出的 FNmRRD (256, 5, 1) 算法相比 LD (16, 20, 5) 获得了 0.34 dB 的增益，且复杂度相当。此外，FNmRRD (16, 5, 1)和 FNmRRD (32, 5, 1)算法分别实现了与 LD (8, 20, 5) 和 LD (16, 20, 5) 类似的性能，但前者使用的迭代次数减少了 90%。

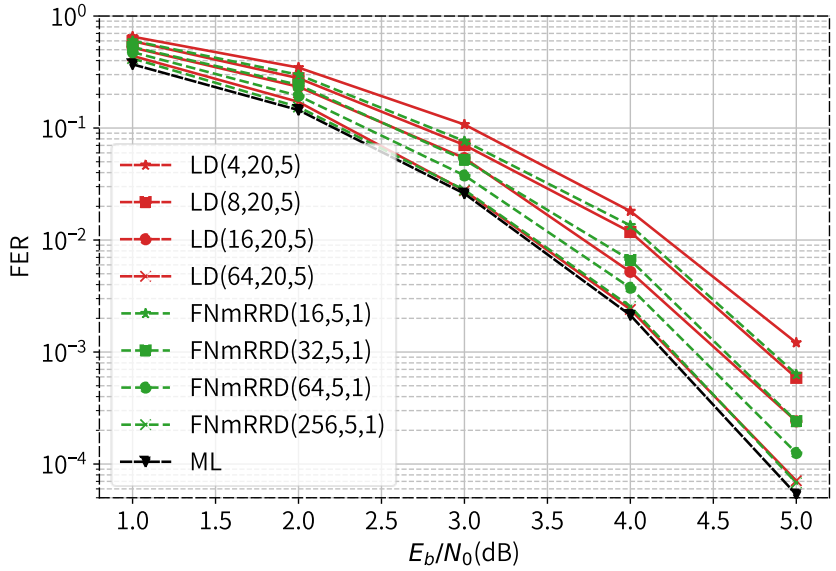

图 4.14 LD 和 FNmRRD 在不同规模下解码 BCH (63,45) 的 FER 性能

Fig. 4.14 FER performance of LD and FNmRRD on the BCH (63, 45) using difference sizes.

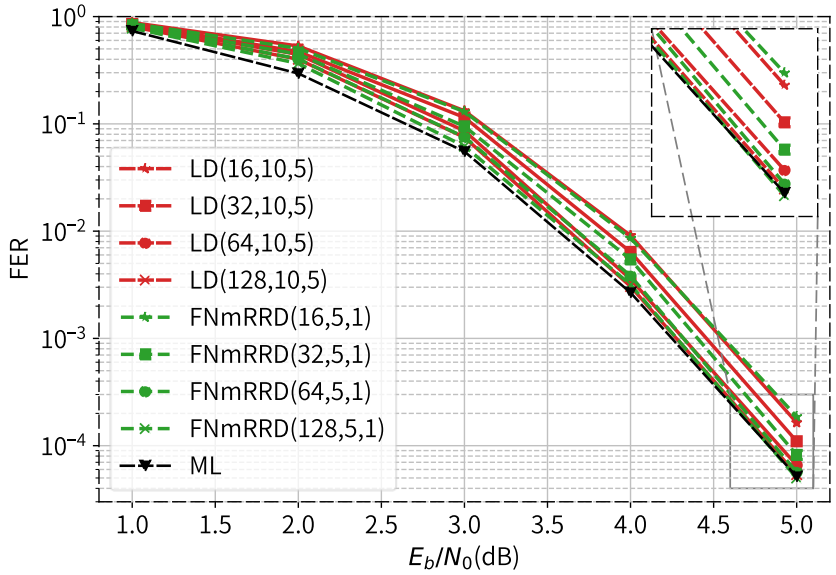

图 4.15 LD 和 FNmRRD 在不同规模下解码 Punctured RM (127, 99) 的 FER 性能

Fig. 4.15 FER performance of LD and FNmRRD on the punctured RM (127, 99) using different sizes.

图 4.15 展示了 LD 和 FNmRRD 对 punctured RM (127, 99) 码进行解码时的 FER 性能。可以看到 LD 算法和 FNmRRD 算法都可以十分逼近 ML 解码。但同宽度下的 FNmRRD 算法的 FER 性能比 LD 算法略好且迭代次数只有 LD 的 10%。特别地，

FNmRRD (64, 5, 1) 的 FER 性能和 LD (128, 10, 5) 的 FER 性能相当但前者所需的最大迭代次数仅为后者的 5%。

对于 QR (47, 24) 码，不同大小的 FNmRRD 算法的 FER 性能如图 4.16 所示。由于 LD 算法需要使用仿射置换对输入进行置换，而仿射置换不能直接应用于 QR 码，所以图中没有展示 LD 的性能。FNmRRD 使用的置换不局限于仿射置换所以不受此影响。由图可知 FNmRRD (128, 5, 1) 算法已经十分接近 ML 解码，当 FER 为10^{-5}时，两者的差距仅为 0.1dB。而本章第 2 小节中我们提出的 Chase-II-mRRD 离 ML 性能尚有 0.5 dB 的差距，且最大迭代次数为$2^5 \times 5 \times 10 \times 5 = 8\,000$，远高于 FNmRRD 的$128 \times 5 \times 1 = 640$。所以我们认为，尽管新提出的 FNmRRD 使用了更加复杂的随机置换，但是其迭代次数却远小于 Chase-II-mRRD（减少超过 90%），并获得了更好的性能。因此，引入更复杂的随机置换是值得的。

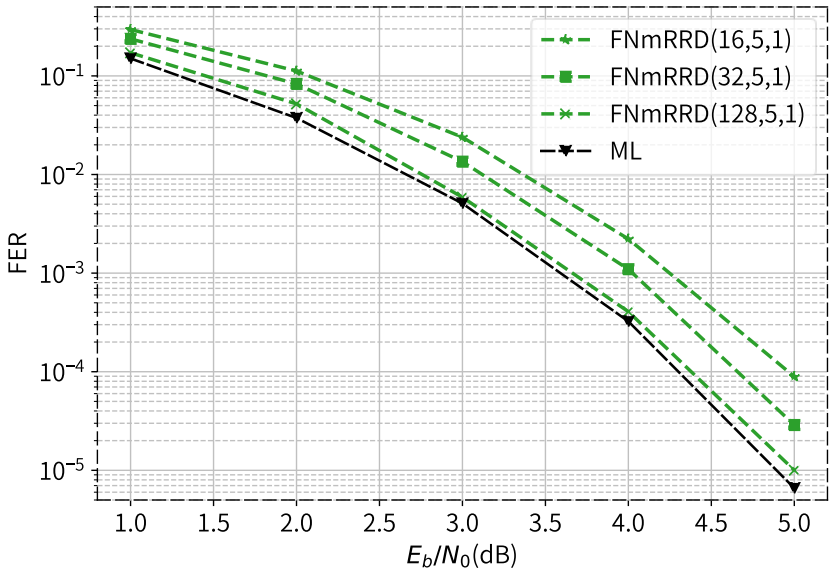

图 4.16 FNmRRD 在不同规模下解码 QR (47, 24)的 FER 性能

Fig. 4.16 FER performance of FNmRRD on the QR (47, 24) using different sizes.

4.4.4 复杂度分析

NLMS 算法和 NmRRD 算法都可以并行运行，但是出于简洁考虑，下文中的复杂度分析都只计算不同操作的总数，也就是串行运行时的复杂度。由前文关于矩阵结构的分析可知，大小为$(n-k) \times n$的$\mathbf{H}$和大小为$n \times n$的$\mathbf{H}_{cyc}$都拥有相同的行重n_r，而对于$\mathbf{H}_{cyc}$而言其列重与行重相等，也为n_r。对于每次迭代，CEBP 解码器需要$n(n_r-1)$次加法，$n(3n_r-2)$次乘法，n次$\tanh(\cdot)$操作和n次$\tanh^{-1}(\cdot)$操作。相比之下，NLMS 每次迭代需要$n(n_r-1)$次加法，$n \times n_r$次乘法，$n(n_r-2)$

次符号翻转以及n次min(·)操作。具体在对 BCH (63, 45) 码进行解码时，所需操作如表 4.2 所示。可以看到，NLMS 算法在加法上与 CEBP 的次数相同，而乘法上只有 CEBP 所耗次数的 1/3 左右。且相比 CEBP 省去了tanh(·)和$\tanh^{-1}(\cdot)$的操作而多出了符号翻转和取最小值的操作。而这两种操作的资源消耗比tanh(·)和$\tanh^{-1}(\cdot)$要小得多。所以总体上 NLMS 每次迭代的计算量小于 CEBP。

表 4.2 NLMS 和 CEBP 解码 BCH (63, 45) 码的单次迭代操作数对比

Table 4.2 Number of operations per iteration for NLMS and CEBP when decode the BCH (63,45).

算法	+	×	tanh(·)	$\tanh^{-1}(\cdot)$	符号翻转	min(·)
CEBP	1 449	4 410	63	63	0	0
NLMS	1 449	1 512	0	0	1 386	63

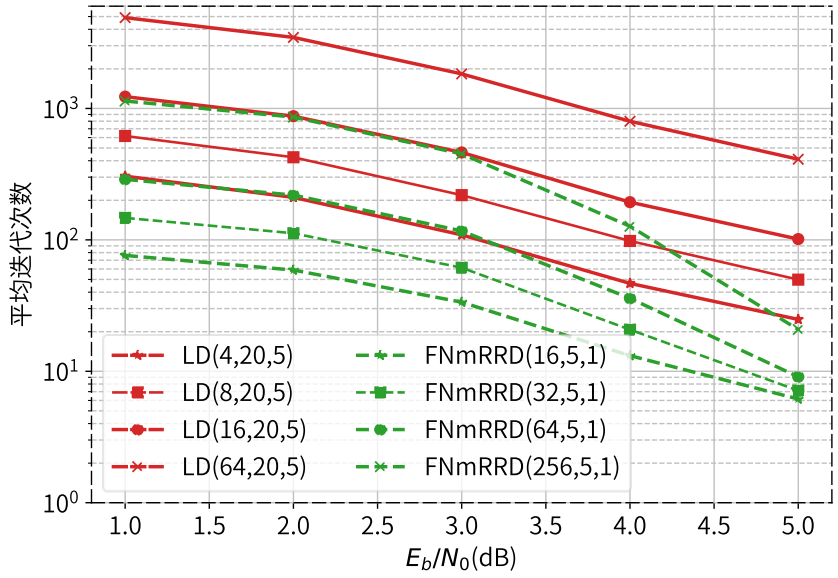

图 4.17 不同规模的 FNmRRD 和 LD 的平均复杂度对比

Fig. 4.17 The average complexity of FNmRRD and LD using different sizes.

除了分析每次迭代的复杂度，我们还将分析 LD 算法和 FNmRRD 算法的平均迭代次数。平均复杂度与前文相同，通过在各个 SNR 取值下仿真 10 000 帧然后计算平均迭代次数得到。在 BCH (63, 45) 码上仿真得到的平均迭代次数结果如图 4.17 所示。前文我们提到，当 FNmRRD 和 LD 在 BCH (63, 45) 码上实现接近于 ML 解码的性能时，前者的最大迭代次数仅为后者的 1/5，图上 0 dB 对应的数据点也验证了我们的分析。此外我们还注意到，随着信噪比的增加，FNmRRD 和 LD 之间迭代次数的差距被进一步放大。例如，当信噪比为 5 dB 时，FNmRRD (256, 5,

1) 的平均迭代次数约为与之解码准确率相当的 LD (64, 20, 5) 的 5%。同样在 5 dB 时，参数规模最大的 FNmRRD (256, 5, 1) 所需的平均迭代次数也比参数规模最小的 LD (4, 20, 5) 算法要略少，而根据图 4.14 此时两者的帧错误率差距约为 18 倍。

4.5 本章小节

本章主要对神经网络循环码解码器进行了多种改动尝试逼近码字的 ML 性能。首先我们提出使用线性规划和整数规划快速获得中短码长码字的 ML 解码性能，以便于后续性能比较。然后我们引入了 Chase-II 算法对第 3 章提出的 NCmRRD 算法进行增强。提出的 Chase-II-mRRD 算法在码长很短时能够较为接近 ML 解码性能，但在码长稍长时与 ML 解码性能相差较大。接着我们通过理论分析和仿真得出结论：第 3 章所使用的神经网络的权重分配策略可以大幅度简化。针对这个问题，本章提出的神经网络解码器采用逐迭代分配权重的方式，大幅度缩减了权重数量。此外我们还引入了神经网络分层 min-sum 算法，同时配合新的权重分配方案大幅度提升了解码准确率并降低了复杂度。本章中我们还对 mRRD 算法进行了修改，得到的算法称为 FNmRRD。尽管 FNmRRD 和 Chase-II-mRRD 算法相比在置换步骤上更加复杂，但带来的收益是解码性能的显著提高以及最大迭代次数的大幅降低。本章还对比了所提出的 FNmRRD 算法和 LD 算法，我们发现前者以比后者更低的最大迭代次数在多个码字上取得了接近 ML 解码的准确率。此外与 LD 算法不同，本章提出的 FNmRRD 算法对码字的置换群没有限制，而 LD 算法只能对置换群包含仿射变换的码进行解码，故 FNmRRD 算法的泛用性更强。最后，本章从每次迭代操作数以及平均迭代次数方面分析了所提出算法的复杂度，发现所提出的 FNmRRD 在每次迭代操作数低于 LD 算法的同时，其平均迭代数远低于后者。

5 总结与展望

5.1 本文内容总结

信道编码使在不可靠的信道上进行信息传输成为可能。在本文中，我们首先介绍了目前移动通信中一个研究热点，即超可靠低延迟通信，从而引入了中短码长的循环码在该场景下的应用前景以及这类编码在应用上目前亟待解决的问题，即缺乏好的解码算法。然后本文介绍了一些循环码的解码算法发展历史，并回顾了目前最先进的基于机器学习的信道解码方法。接着就神经网络 BP 类算法这个子领域展开了深入的探讨和研究。本文从两个方面着手改进神经网络 BP 类算法，以更好地对循环码进行解码：提升神经网络解码器的准确率，降低神经网络解码器的复杂度。本文的贡献可以概括为以下三个方面：

（1）对影响神经网络 BP 解码器性能的因素进行了研究。神经网络 BP 类解码器提高解码准确率的机制可能有三点：通过收缩网络上信息传递的幅度来拟合正确的 LLR 幅度、通过给环上的边分配不同权重来减轻环对解码的影响、通过给陷阱集中的边分配权重来减轻陷阱集对解码的影响。我们的研究表明其中的第二点在神经网络解码器中的作用占主导地位。此外我们发现四环对于高密度循环码的影响与其对 LDPC 码的影响不同。BP 类算法用于对高密度循环码解码时，在四环多的地方产生的错误不一定会比在四环少的地方产生的错误多。和 LDPC 类似，当神经网络 BP 类算法用于解码循环码时，小型陷阱集是影响其准确率的最重要因素。

（2）针对陷阱集，我们提出引入 mRRD 算法将比特从陷阱集中置换出去，并将其他比特引入陷阱集，从而降低误码率。结合 mRRD 和神经网络 TAMS 解码器后所得到的神经网络 mRRD 算法在解码准确率上超过了现有的 DS 算法。同时，相比 DS 算法，mRRD 拥有更低的渐近复杂度。此外，我们针对 mRRD 在宽度较小的情况下进行了简化，由此得到的神经网络循环 mRRD 相比原神经网络 mRRD 算法性能损失很小，但复杂度得到大幅度简化。由于无需求解置换群且使用的置换只包含循环移位，该算法更易于在电路上实现。在目标编码的置换群难以求解或者置换群极大时，mRRD 算法很难运行，而神经网络循环 mRRD 则不受影响。此外，我们引入了 Chase-II 算法并与神经网络循环 mRRD 算法进行结合，所提出的 Chase-II-mRRD 算法在短码上可以较为接近 ML 解码。

（3）为了进一步提升性能以接近 ML 解码，我们引入了神经网络分层 min-sum 算法。该算法拥有更高的收敛速度，在同样的迭代次数下可以获得更高的准确率。基于①的分析，我们在不影响解码准确率的情况下大幅度削减了网络权

重的数量，使得训练速度更快、解码的复杂度更低，并使得解码器可用于更长的码字。最后，我们通过在 mRRD 中应用快速终止条件，降低了其平均复杂度从而能在复杂度可以接受的情况下增大其规模。最终在多种码上逼近了 ML 解码的性能。仿真结果表明：我们提出的 FNmRRD 解码器和目前最先进的神经网络 LD 解码器相比，拥有同样的解码准确率，但算法复杂度要远低于后者。

本文所提出的算法在多个不同码长的 QR 码上均获得了优异的性能，且提出的算法同样可以应用于其他类似的循环码上。值得一提的是，本文最后提出的 FNmRRD 在 QR (47, 24)、BCH (63, 45)和 punctured RM (127, 99) 码上都获得了十分逼近 ML 解码的准确率。

5.2 未来工作展望

本文的工作主要围绕提升神经网络 BP 类算法解码循环码的准确率以及降低相应的解码复杂度这两方面展开。所提出的方法在 QR 码、BCH 码、和 punctured RM 码都展示出优异的性能。但是这并不代表在这方面的研究已经完善，后续还可以从以下几个方面进一步开展研究：

（1）探索四环影响神经网络 BP 类算法解码循环码的深层机制。众所周知，无论环是否构成了小型陷阱集，BP 类算法会由于图中有短环而性能劣化。这一点在 LDPC 码中尤为明显，现有构建 LDPC 码的算法都会极力避免四环的出现。而在本文的实验中，错误并未集中在四环密度高的地方。相反地，四环密度高的地方错误频数反而更低，这可能与循环码的一些特性有关，仍需进一步研究。

（2）研究不同的节点更新机制。本文只考虑了泛洪传播以及分层传播两种方法。目前学界已经提出了其他更高效的传播调度机制。例如基于 LLR 残差的动态调度机制等等[49]-[50]。由于时间和篇幅所限，本文并未对动态调度机制和神经网络之间的结合做出研究，采用动态调度机制的神经网络解码器可能拥有更加优越的性能。

（3）应用不同的冗余校验矩阵。本文的神经网络主要针对普通的校验矩阵$\mathbf{H}$以及循环冗余校验矩阵$\mathbf{H}_{cyc}$进行设计，而文献[36]1957-1966 中提到使用对偶码的所有具有最小重量的码字构造冗余校验矩阵进行解码。该方法对于码长偏长的码字实用度较低，但是结合前文中的结论，即小型陷阱集是影响神经网络 BP 类算法解码性能的最主要因素，未来的研究可以考虑通过引入对偶码的最小重量码字或使用其他方法对校验矩阵进行扩充从而扩大小型陷阱集，同时兼顾算法的复杂度和 Tanner 图中陷阱集的大小。

（4）最后，出于简洁性考虑，本文的通信模型仅局限于 BI-AWGN 信道，这是一个较为简单理想的信道。然而现实中的信道模型则更为复杂，未来的研究工

作可以考虑神经网络在不同调制模式以及不同信道例如擦除信道、突发错误以及衰落信道等场景下的应用和优化

。

参考文献

[1] Popovski P St f nović Č, Nielsen J J, et al. Wireless access in ultra-reliable low-latency communication (URLLC)[J]. IEEE Transactions on Communications, 2019, 67(8): 5783-5801.

[2] Gallager R G. Low-density parity-check codes[J]. IRE Transactions on Information Theory, 1962, 8(1): 21-28.

[3] Arikan E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels[J]. IEEE Transactions on Information Theory, 2009, 55(7): 3051-3073.

[4] Wang F, Zhang J J, Zheng X, et al. Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond[J]. IEEE/CAA Journal of Automatica Sinica, 2016, 3(2): 113-120.

[5] Floridi L, Chiriatti M. GPT-3: Its nature, scope, limits, and consequences[J]. Minds and Machines, 2020, 30(4): 681-694.

[6] Thoppilan R, de Freitas D, Hall J, et al. Lamda: Language models for dialog applications[EB/OL]. [2022-02-10] https://arxiv.org/abs/2201.08239.

[7] Song Y, Jascha S, Kingma D P, et al. Score-based generative modeling through stochastic differential equations[EB/OL]. [2021-02-10] https://arxiv.org/abs/2011.13456.

[8] Prange E. Some cyclic error-correcting codes with simple decoding algorithms[R]. Cambridge, MA, USA: Air Force Cambridge Research Laboratory, 1985.

[9] Berlekamp E R. Algebraic coding theory [M]. Revised ed. Singapore: World Scientific Publishing Co. Pte. Ltd., 1984.

[10] Blahut R E. The Gleason-Prange theorem[J]. IEEE Transactions on Information Theory, 1991, 37(5): 1269-1273.

[11] MacWilliams F J, Sloane N J A. The theory of error-correcting codes[M]. Amsterdam: North-Holland Publishing Company, 1977.

[12] Wicker S B. Error control systems for digital communication and storage[M]. New Jersey: Prentice-Hall, Inc., 1994.

[13] Honary B, Hunt B, Maundrell M. Improving automatic link establishment through a new soft decision trellis decoder for the (24, 12) Golay code[C]//University of York. Sixth International Conference on HF Radio Systems and Techniques. York, UK: Institution of Electrical Engineers, 1994: 182-185

[14] Li Y, Duan Y D, Chang H C, et al. Using the difference of syndromes to decode quadratic residue codes[J]. IEEE Transactions on Information Theory, 2018, 64(7): 5179-5190.

[15] Bose R C and Ray-Chaudhuri D K. "On a class of error correcting binary group codes[J]. Information and control, 1960, 3(1), 68-79.

[16] Hocquenghem A. Codes correcteurs d'erreurs[J]. Chiffres, 1959, (2), 147-156.

[17] Massey J. Shift-register synthesis and BCH decoding[J]. IEEE Transactions on Information Theory, 1969, 15(1): 122-127.

[18] Muller D E. Application of Boolean algebra to switching circuit design and to error detection[J]. Transactions of the IRE professional group on electronic computers, 1954 (3): 6-12.

[19] Reed I S. A class of multiple-error-correcting codes and the decoding scheme[J]. Transactions of the IRE Professional Group on Information Theory, 1954, 4(4):38-49.

[20] Kim H, Jiang Y, Rana R B, et al. Communication Algorithms via Deep Learning[C/OL]. International Conference on Learning Representations (ICLR), 2018, [2023-04-20]. https://openreview.net/pdf?id=ryazCMbR-.

[21] Zhang Y, Wu H, Coates M. On the design of channel coding autoencoders with arbitrary rates for ISI channels[J]. IEEE Wireless Communications Letters, 2021, 11(2): 426-430.

[22] Jiang Y, Kim H, Asnani H, et al. Learn codes: Inventing low-latency codes via recurrent neural networks[J]. IEEE Journal on Selected Areas in Information Theory, 2020, 1(1): 207-216.

[23] Jiang Y, Kim H, Asnani H, et al. Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels[C]//NeurIPS. Advances in Neural Information Processing Systems 32. Vancouver, BC, Canada: NeurIPS Foundation, Inc., 2019: 2758-2768.

[24] Huang L, Zhang H, Li R, et al. AI coding: Learning to construct error correction codes[J]. IEEE Transactions on Communications, 2019, 68(1): 26-39.

[25] Gruber T, Cammerer S, Hoydis J, et al. On deep learning-based channel decoding[C]//IEEE. 51st Annual Conference on Information Sciences and Systems (CISS). Baltimore, MD, USA: IEEE, 2017: 1-6.

[26] Nachmani E, Be'ery Y, Burshtein D. Learning to decode linear codes using deep learning[C]//IEEE. 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton). Pisacataway, NJ, USA: IEEE, 2016: 341-346.

[27] Lugosch L, Gross W J. Neural offset min-sum decoding[C]//IEEE. 2017 IEEE International Symposium on Information Theory (ISIT). Aachen, Germany: IEEE, 2017: 1361-1365.

[28] 郭军军, 白硕栋, 王乐. 基于深度学习的实用 HDPC 码译码方法研究[J]. 计算机系统应用, 2019, 28(4): 247-251.

[29] Nachmani E, Marciano E, Lugosch L, et al. Deep learning methods for improved decoding of linear codes[J]. IEEE Journal of Selected Topics in Signal Processing, 2018, 12(1): 119-131.

[30] 何彦琦, 彭大芹, 赵雪志. 一种基于循环神经网络的极化码 BP 译码算法[J]. 计算机工程,

2022, 48(1): 197-203.

[31] Dimnik I, Be'ery Y. Improved random redundant iterative HDPC decoding[J]. IEEE Transactions on Communications, 2009, 57(7): 1982-1985.

[32] Nachmani E, Wolf L. Hyper-graph-network decoders for block codes[C]//NeurIPS. Advances in Neural Information Processing Systems 32. Vancouver, BC, Canada: NeurIPS Foundation, Inc., 2019: 2329–2339.

[33] Chen X, Ye M. Cyclically Equivariant Neural Decoders for Cyclic Codes[C]//International Conference on Machine Learning (ICML). Proceedings of the 38th International Conference on Machine Learning. Cambridge, MA, USA: PMLR, 2021: 1771-1780.

[34] Chen X, Ye M. Improving the List Decoding Version of the Cyclically Equivariant Neural Decoder[C]//IEEE. 2022 IEEE International Symposium on Information Theory (ISIT). Espoo, Finland: IEEE, 2022: 2344-2349.

[35] Chen X, Ye M, Neural Decoders with Permutation Invariant Structure[J], Journal of the Franklin Institute, 2023, doi: https://doi.org/10.1016/j.jfranklin.2023.03.024

[36] Buchberger A, Häger C, Pfister H D, et al. Pruning and quantizing neural belief propagation decoders[J]. IEEE Journal on Selected Areas in Communications, 2020, 39(7): 1957-1966.

[37] Hamming R W. Error detecting and error correcting codes[J]. The Bell system technical journal, 1950, 29(2): 147-160.

[38] Tanner R. A recursive approach to low complexity codes[J]. IEEE Transactions on information theory, 1981, 27(5): 533-547.

[39] Hatami H, Mitchell D G M, Costello D J, et al. A threshold-based min-sum algorithm to lower the error floors of quantized LDPC decoders[J]. IEEE Transactions on Communications, 2020, 68(4): 2005-2015.

[40] Richardson T. Error floors of LDPC codes[C]// University of Illinois at Urbana-Champaign. Proceedings of 41st Annual Allerton Conference on Communication. University of Illinois at Urbana-Champaign, 2003: 1426-1435.

[41] Karimi M, Banihashemi A H. Efficient Algorithm for finding dominant trapping sets of LDPC codes[J]. IEEE Transactions on Information Theory, 2012, 58(11): 6942-6958.

[42] Sims C C. Computational methods in the study of permutation groups[C]// Atlas Computer Laboratory. Computational problems in abstract algebra. Oxford, UK: Pergamon Press, 1970: 169-183.

[43] Tieleman T and Hinton G. “N ur l n tworks for m hin l rning (Lecture 6a) ” [EB/OL]. [2023/04/22]. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[44] Feldman J, Wainwright M J, Karger D R. Using linear programming to decode binary linear

codes[J]. IEEE Transactions on Information Theory, 2005, 51(3): 954-972.

[45] Zhang X, Siegel P H. Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes[J]. IEEE Transactions on Information Theory, 2012, 58(10): 6581-6594.

[46] Chase D. Class of algorithms for decoding block codes with channel measurement information[J]. IEEE Transactions on Information Theory, 1972, 18(1): 170-182.

[47] Kaneko T, Nishijima T, Inazumi H, et al. An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder[J]. IEEE Transactions on Information Theory, 1994, 40(2): 320-327.

[48] Elidan G, McGraw I, Koller D. Residual belief Propagation: informed scheduling for asynchronous message passing[C]//AUAI. Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence. Arlington, Virginia, USA: AUAI Press, 2006: 165-173.

[49] Casado A I V, Griot M, Wesel R D. Informed dynamic scheduling for belief-propagation decoding of LDPC codes[C]//IEEE. 2007 IEEE International Conference on Communications: Glasgow, Scotland, 24-28 June 2007. Piscataway, NJ, USA: IEEE, 2007: 932-937.

[50] Jing Y, Zhang W, Wang H, et al. Improved Adaptive Belief Propagation Decoding of Reed-Solomon Codes With SPC Codes[J]. IEEE Communications Letters, 2022, 26(7): 1464-1468.

B. 学位论文数据集

关键词	密级	中图分类号
循环码；深度神经网络；信念传播算法；迭代解码；最大似然解码	公开	TP

学位授予单位名称	学位授予单位代码	学位类别	学位级别
重庆大学	10611	学术学位	硕士

论文题名	并列题名	论文语种
循环码的神经网络信念传播解码算法研究		中文

作者姓名	王铭	学号	202014021072T
培养单位名称		培养单位代码	
重庆大学		10611	
学科专业	研究方向	学制	学位授予年
计算机科学与技术	机器学习	3 年	2023
论文提交日期	2023 年 6 月	论文总页数	69
导师姓名	黎勇	职称	教授
答辩委员会主席		李传东	

电子版论文提交格式

文本（√） 图像（） 视频（） 音频（） 多媒体（） 其他（）