可变码长的极化码编译码算法研究

摘 要

Arikan 在 2009 年第一次提出了极化码，而且在理论上严格证实了极化码在二进制输入对称离散无记忆信道下，能够“实现”香农容量，这对于我们实现大容量通信传输的信道编解码技术有重要意义，并且极化码的编译码算法都具备较低的复杂度，性能在一定码长下也非常优秀，因此在它面世之后，就吸引到了充分重视并得到了普遍应用。然而在实际的应用中，传输信道的状态在变化，为了适应这种变化，就需要对极化码的码长、码率等参数进行灵活调整，因此必须对可变码长的极化码的编译码算法进行研究。本文以此为背景，分析了当前可变码长的极化码编译码方案存在的问题，针对于编码方面调整码长后极化码性能下降的情况，提出了一种可变码长的极化码编码算法。在传统调整极化码码长的凿孔方法下，再基于信道可靠性估计的方法来计算各个极化子信道的错误概率，以此来将可靠性程度比较低的极化子信道凿掉。并且为了保证了更好的性能，将被凿掉比特的初始对数似然比设置为无穷大(或负无穷大)。仿真结果表明，本文所提出的改进后的编码算法具有比传统的编码算法更好的性能表现。

对于改进后的编码算法，为了使其发挥更好的性能，本文将深度神经网络技术引入到可变码长的极化码译码问题的研究上。深度学习技术日渐成熟稳定，其针对处理和解决分类问题存在十分良好的作用功能体现，而极化码的信道译码问题，从在实际中而言，同时也是高维空间方面的分类问题，所以使用实际深度神经网络技术处理和解决译码问题，逐渐发展成为当前研究的核心热点。基于这个方面，本文针对几种常见的深度神经网络，在不规则码长下对它们的译码性能进行了比较分析。针对译码系统结构，本文提出了一种可变码长的极化码译码算法，通过改变系统框架达到改变学习目标的目的从而达到更好的译码性能。传统类型的神经网络译码体系要求神经网络不仅需要学习噪声特征，还需要学习码字处到数据信息位置的码字结构映射。通过研究分析能够知道，码字结构在实际中是 N 个比特校对核验问题的分布组成集合，这是实际深度神经网络无法高效学习的。本文经过改变神经网络译码器的学习目标，提出了关于可变码长的极化码神经网

络译码结构，规避了神经网络被动式学习码字结构映射，并且应用泛化能力非常强的神经网络，展开了可变码长极化码的译码学习。仿真实验的最终处理结果说明，改善后的神经网络泛化能力获得了很大的提高，仅仅需要学习码本空间方面的小部分码字，就能够主动学习到码字的空间分布特征，并且能实现接近于最大后验概率译码算法的译码性能。

关键词：

信道编码， 极化码， 编译码算法， 凿孔， 深度神经网络

Research on Polar Code Encoding and Decoding Algorithm with Variable Code Length

ABSTRACT

Arikan first proposed polar codes in 2009 and provided a rigorous theoretical proof that polar codes can "achieve" Shannon capacity under binary input symmetric discrete memoryless channels . This has significant implications for channel coding and decoding techniques in realizing high-capacity communication transmission. Polar codes exhibit relatively low complexity in encoding and decoding algorithms, and their performance is excellent within certain code lengths. Consequently, since their introduction, they have attracted considerable attention and found widespread application. However, in practical applications, the channel conditions change. To adapt to these changes, it is necessary to flexibly adjust parameters such as the code length and code rate of polar codes. Therefore, it is essential to study encoding and decoding algorithms for variable-length polar codes. In this context, the present study analyzes the existing problems in the current variable-length polar code encoding and decoding schemes, and proposes an improved variable-length polar code encoding algorithm to address the performance degradation of polar codes after adjusting the code length. Under the traditional puncturing method for adjusting the polar code length, the error probability of each polar sub-channel is calculated based on the channel reliability estimation method, and the polar sub-channels with relatively low reliability are punctured. To ensure better performance, the initial log-likelihood ratio of the punctured bits is set to infinity (or negative infinity). Simulation results show that the improved encoding algorithm proposed in this paper has better performance than the traditional encoding algorithm.

To further improve the performance of the encoding algorithm, this paper introduces deep neural network technology into the research of variable-length polar code decoding. Deep learning technology is increasingly mature and stable, demonstrating excellent performance in addressing classification problems. Polar code channel decoding is a high-dimensional classification problem in practice, making deep neural network technology suitable for handling and solving decoding issues, and has

gradually become the focus of current research. Based on this, the study compares and analyzes the decoding performance of several common deep neural networks under irregular code lengths. In terms of decoding system structure, this paper presents a variable-length polar code decoding algorithm that achieves better decoding performance by changing the learning objectives through modifying the system framework. Traditional neural network decoding systems require the neural network to learn not only the noise characteristics but also the codeword structure mapping from the codeword position to the data information position. Research and analysis reveal that the codeword structure is a distribution composed of N-bit parity-check problems in practice, which is difficult for deep neural networks to learn efficiently. By altering the learning objective of the neural network decoder, a variable-length polar code neural network decoding system structure is proposed, which avoids forcing the neural network to learn the codeword structure mapping and applies the highly generalizable neural network to conduct variable-length polar code decoding learning. The final simulation results show that the improved neural network's generalization ability has been significantly enhanced, requiring only a small part of the codebook space to independently learn the spatial dispersion characteristics of the codewords and achieve performance close to the MAP decoding algorithm.

Keywords:

Channel coding, Polar codes, Coding and decoding algorithms, Puncturing scheme, Deep neural networks

关于学位论文使用授权的声明

本人完全了解吉林大学有关保留、使用学位论文的规定，同意吉林大学保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权吉林大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

（保密论文在解密后应遵守此规定）

论文级别： ☑硕士 □博士

学科专业： 信息与通信工程

论文题目： 可变码长的极化码编译码算法研究

作者签名： 韩天威 指导教师签名：

2023 年 5 月 28 日

目　录

第 1 章　绪论

1.1　研究背景及意义

通信技术发展日新月异，产生了大量的实际应用，使我们的社会变成为一个万物互联的社会。它的发展已经对社会产生了深远的影响，截止到 2020 年，有超过 500 亿台通信连接设备，它们在医疗保健、自动驾驶、制造业、农业、航运和娱乐等行业中发挥着巨大的作用。但人们认为，最大的影响尚未到来。预测至 2035 年全世界经济收益总产出实现 12.3 万亿美金，2036 年，通讯经营公司将会在全世界雇佣 2200 万务工人员。5G 就是其中一个例子，它目前有三个不同的应用：（1）eMBB-加强移动信号宽带；（2）URLLC-超安全可靠的低推迟作用通讯；（3）mMTC-大型机器种类通讯。这三个使用都有不同的要求，从 eMBB 的高达 10Gbps 的速率到 URLLC 少于 10e-5 丢包，以及通过 mMTC 设想的超过 500 亿个低功耗设备。这些巨大的不同的需求和规范对适合每个应用的标准和算法提出了极高的要求。

现代数字通信的核心是信道编码，或者也称为前向纠错（Forward Error Correction,FEC），根据未来通信系统的条件，信道编码还必须满足如高速率、超高可靠性以及低功耗等要求。信道编码领域自诞生之日起就对社会产生了深远影响。在 70 年代末，Ungerboeck 指出了协同调节控制和程序代码设计方案—格子程序代码调节控制（Trellis-Coded Modulation）。这个方案能够将普通电话线上的调制解调器的吞吐量从 23kbps 提高到 45kbps，这被认为是在互联网早期阶段使其更加广泛的一个重要因素，例如使在互联网上下载媒体文件成为可能。

Arikan 在 2009 年第一次提出了极化码[1]，而且在理论上充分证实了极化码在二进制输入对称离散无记忆信道下，就能够“实现”香农容量，这对于我们实现大容量通信传输的信道编解码技术有重要意义，并且极化码的编译码算法都具备较低的复杂度，性能在特定码长下也非常优秀，因此在它面世之后，就吸引了充分重视并得到了普遍应用。然而在实际的应用中，传输信道的状态在变化，为了适应这种变化，我们就需要对极化码的码长、码率等参数进行灵活调整，因此，我们必须对极化码的编码算法进行研究。

第五代移动通信对通信传输的高品质要求，决定了其对于高速率、高可靠、低时延的译码器的需求。对于极化码，即使传统类型的译码算法及改善过的算法都具备了非常好的译码性能，例如 SC 译码算法及其改进译码算法，然而在译码时延上受制于译码算法本身，无法实现低时延译码。所以，设计一种全新的高速率、低时延、高可靠的译码算法，逐渐发展为未来通信的研究核心重点。最近几年深度学习技术的研究发展和大规模推广利用，为通信领域范围提供了全新的方向和技术来处理和解决上述问题。深度学习技术在分类问题的使用上，存在十分好的应用功能效果，而通信里的信道译码问题，在实际中被理解为分类问题，以极化码的信道译码为例，针对码长是 N、数据信息比特数目是 K 的极化码 $P(N,K)$，其在 N 维分布空间里具有 2^K 个码字种类，则译码问题能够描述为将接收序列分类到 2^K 个码字类别里对应的正确码字类别。所以，将深度学习技术应用在通信中的信道译码问题是可行的。

1.2 可变码长的极化码

1.2.1 可变码长的极化码删余方案

我们通常通过采用删余方法来调整极化码的恒定不变码长。三种重要的极化码删余策略包含：第一，转变码字组成结构矩阵[2]的渠道，虽然这样，极化码的码长依然受制于 $N = L^n$（n=1，2，…），L 代表组成结构矩阵的分布维度；第二，将极化码和 LDPC 或者其他数学线性具体分组码相结合，形成一种级联码[3-4]的方式，由于极化码和其他具体分组码的编译码模式存在差异，使得级联码的编译码复杂性较高；第三，通过删除极化码生成矩阵的指定行或者列，产生一类全新的数学线性具体分组码的凿孔技术。这类具体分组码不会直接干扰影响极化码的编译码结构，所以保存了极化码编译码低复杂度的特征，是一类简单好用的极化码改变规则码长的设计方案。本次深入研究分析主要应用第三种模式对极化码进行凿孔，以实现调整规则码长的目标。

1.2.2 可变码长的极化码编译码算法研究现状

参考文献[5]第一次对极化码凿孔策略进行了研究，提出了随机凿孔设计方案和停止树凿孔设计方案。随机凿孔设计方案是对编码后的码字比特随机选用展

开凿孔，具有较大的随机性，性能波动较大，可能产生良好译码作用效果，也会完全没有办法编译码。停止树凿孔设计方案模仿参考了 LDPC 码 Tanner 流程设计图的组成构造理论，这样只可以运用和 Tanner 流程设计图相互对应的置可信度传播（Belief Propagation,BP）译码器，展开了编译码，其他如串联相消（Successive Cancellation,SC）、串联相消列表（Successive Cancellation List,SCL）及串联相消栈（Successive Cancellation Stack,SCS）等译码器效果不佳，限制了其实际应用范围。尽管上述两种凿孔方案在特定条件下可行，但没有充分利用极化码的递归特性，也非最优凿孔策略。参考文献[6]采用穷搜索法指出了根据简约化极化矩阵的凿孔设计方案，不需要修整译码器结构，确保了极化码的极化作用效果最理想化，而且作用功能领先于参考文献[5]中指出的随机凿孔设计方案。但是因为应用了穷搜算法，其搜查复杂度超过参考文献[5]；同时，需要额外算法计算简化极化矩阵的输入端冻结序列，实际编码结构计算量较大，不利于实现。文献[7]证实，寻找最优码字端凿孔方式等于第一步发现数据通讯消息序列端，其最合理凿孔方式，之后经过映射相互关系运算出实际码字端凿孔模式。该文献提出了一种启发式算法，先选择数据通讯消息序列端凿孔方式，再经过映射相互关系获取真实码字端凿孔比特具体位置，但是忽视了凿孔对其余信号通道的作用影响，造成极化作用效果损失。参考文献[8]指出了一类准匀凿孔（Quasi Uniform Puncturing,QUP）算法，实验结果显示，在 WCDMA 或 LTE 无线通信系统中，凿孔后的极化码的作用功能领先于 Turbo 码。但是，QUP 算法在译码端缺乏相互对应凿孔比特的先验数据信息，译码端初始对数似然比会直接干扰影响数据信息比特，码率愈大，QUP 算法对数据信息比特的作用影响也愈大，因此一般应用在低码率基本条件。参考文献[9]在 QUP 算法基础应用之上，指出了一类低复杂度的极化码凿孔算法。但是，减少复杂度是以牺牲极化码的作用功能为经济代价的。该设计方案中，凿孔具体位置分散反过来严格限制了数据通讯消息序列端中数据通讯消息比特具体位置集合，当凿孔数改变的时候，译码器结构大概率会要求相互对应调整。为了在保存凿孔极化码极化作用效果的相同时刻，减少搜查复杂度，刘荣科等人指出了一类创新算法[10]。该算法采用基于权重为 1 的列的方法构建新的凿孔极化码。凿孔位置在译码器已知的情况下，将被凿去比特的有效数值在编码时，调整为固定数值 0 或者 1，译码端补充其对数似然比（Log Likehood Ratio,LLR）是正无穷或者负无穷。在高码率影响作用环境里，此模式译码端初始

LLR 对数据信息比特的作用影响，低于经常使用的 QUP 算法，并且能最大作用程度地保存极化码的极化作用效果，因此这种新型的凿孔方案[10]得到了大家的普遍认可，广泛应用在极化码的速率匹配之中。文献[11]将极化理论推广到非均匀信道，在加性高斯白噪声(Additive White Gaussian Noise,AWGN)信道和 SC 译码的情况下，它可以实现任意码长极化码的构造。但是，这种方法只针对于码率在 1/2 以下的时候才可以取得比较好的性能。文献[12]研究了穿孔位置对固定信息集的影响，提出了一种信息集近似刺穿算法，该算法在已知信息位之前设置一定数量的保护位，然后根据实际传输块长度计算所需的凿孔模式，同样的，这种方法适用于码率在 1/2 以下的情况才可以取得比较好的性能。对于较高码率的情况来说，文献[13]对于每个信息位，定义了一个称为可靠性评分的度量，以使最小可靠性分数最大化的方式来决定被凿孔的位置。但这种方法是有限制条件的，算法中采用不同的权重因子α会有不同的性能表现。

在信道极化之后，较劣的子信道对极化码的误码表现产生关键作用。因此，在凿孔过程中，确定策略将最糟糕的子信道纳入凿孔位是提高凿孔极化码性能的核心问题。然而，凿孔方案[10]并未针对此问题进行改进。其构造流程是直接删去生成矩阵的后 P 行及相应列，并将长度为 N 的极化码最后 P 位设置为冻结比特，从而获得长度为 M 的凿孔极化码。在对生成矩阵进行相应列删除时，若遇到权重相同的列，无法有效挑选相对较差的子信道进行删除。本文参考文献[10]的凿孔方法，但在凿孔策略中引进了信号通道可行性评测模式[11]。运算过程中经过运算每一个极化子信号通道的错误几率，以评估每一个极化子信号通道的安全可靠作用程度，进而将选入凿孔分布矢量的列一一对照的安全可靠度偏低的子信号通道增添到冻结集中。仿真模拟最终分析结果可知，这类模式从某种层面里能够提升所获取的凿孔极化码的性能。

1.3 本文主要研究内容及结构安排

第 1 章是绪论，主要论述了本文的研究背景，从而引出本文的研究价值。简要介绍了几种改变极化码规则码长的方法，分析了当前提出的多种极化码凿孔方案，对比了不同方案的编译码算法的性能。

第 2 章介绍了极化码的基本理论，这是我们后续研究的基础。其中包括几种常见的信道纠错码，从而引出了我们的主要研究对象：极化码，而后本文介绍了

极化码编译码算法所必需的基本信息论知识和进行极化码编译码算法仿真所用到的方法。编码方面，详细阐述了极化码的构造原则，其中包括信道极化原理和几种常见的编码算法；译码方面，详细阐述了几种常见的译码算法，分析了它们的基本原理和运行规则，对比分析了它们的性能。

第 3 章对可变码长的极化码的编码算法进行了研究。首先介绍了传统的极化码凿孔方案，并且详细分析了几种信道可靠性估计的方法，并用他们进行了仿真对比它们在评估分裂子信道可靠性程度方面的优劣。最后将一种改进后的信道可靠性估计方法引入到评估凿孔位上，提出了一种基于凿孔技术的极化码编码算法，分析对比了我们提出的算法和其他几种编码算法的性能。

第 4 章对可变码长的极化码的译码算法进行了研究。首先基于不规则码长的极化码，在几种常见的深度神经网络下进行译码性能分析比较。而后改变系统结构框架，经过转变学习目标，让神经网络只学习接收序列到码字端口的译码映射，进而规避了码字结构映射的学习。考虑到神经网络存在非常强的学习和泛化能力，根据改进的体系结构，我们提出了根据学习噪声的神经网络译码器。仿真实验的最终分析结果可知，改善的神经网络泛化能力获取了很大的提高，仅仅要求学习码本空间方面的一小部分码字，就能够完成接近于最大后验概率(Maximum a Posteriori Probability, MAP)译码算法的性能。

第 5 章为总结与展望。本章概括了本文的所有内容和主要贡献，同时提出了论文工作的不足和未来需要深入探讨的领域。

第 2 章　极化码基本理论

在本章中，我们将讨论构造极化码所需的一些基础知识以及极化码的基本编译码算法。

2.1　极化码的基础知识

2.1.1　信道前向纠错编码

数据从一个地方传输到另一个地方的一个非常基本的部分是通信通道。该通道基本上是传输数据的介质，例如电缆、空气等。由于这些介质不是数据传输的理想介质，它们总是含有噪声的。因此，接收方看到的数据可能与发送方发送的数据不相同，从而导致数据传输不可靠。为了提升数据信息自动输送的可行性，应用了偏差问题测试和纠正错误技术。错误检测是确定接收到的数字数据是否确实是发射机发送的数据的过程。这种编码方案的一个非常简单的例子是重复码，它通过信道发送数据块的多个副本。例如，一个 0111 的块可以被复制三次，然后发送为 011101110111。如果接收方接收到 010101110111 的数据，它可以判断该数据包含错误，因为第一个数据块与后面两个数据块不相同。它还可以通过选择更频繁发生的块来恢复数据，在本例中是 0111。这个操作叫做纠错。然而，这种编码方法的效率非常低。为了能够可靠地重建传输数据，使用了纠错码。

FEC 是一种将冗余数据(奇偶校验位)添加到信息(编码)中，并将增强的数据通过信道发送的方法。在接收端，接收端使用奇偶校验位(译码)重建数据。FEC 的优点是在发生错误时不需要反向信道与发射机通信。因此，它更为简单，可用于广泛的通信渠道。基本的信道编码结构如图 2.1 所示。我们通过一个有噪声的信道发送由 K 比特描述的信息 X，我们想通过引入某种类型的冗余来恢复这个信号，从而使我们最终得到 $N>K$ 。

图 2.1 信道编码

我们的目标是找到代码或算法，1)允许我们以低错误概率恢复信息，2)减少

我们通过信道发送的冗余量，3)在可接受的计算复杂度下实现上述两者。

然而，问题是可以在信息中添加多少奇偶校验位才能使通信可靠。香农定理可以回答这个问题，香农在 1948 年对外公开发表的具备里程碑影响意义的论文中指出，如果我们允许足够长的码长度，我们可以实现错误率极低的速率，这个可以实现的最佳速率被称为信道容量 C。该定理指出，在已知错误概率或信噪比的信道上可以建立可靠通信的最大信息速率。这个最大信息速率称为通道容量。

自从香农表明在有噪声信道上可实现的信息速率有一个上限以来，编码理论家们一直在努力寻找实现信道容量的有效方法，因为香农的工作并没有真正给出任何关于如何实现容量或甚至接近容量的方法。

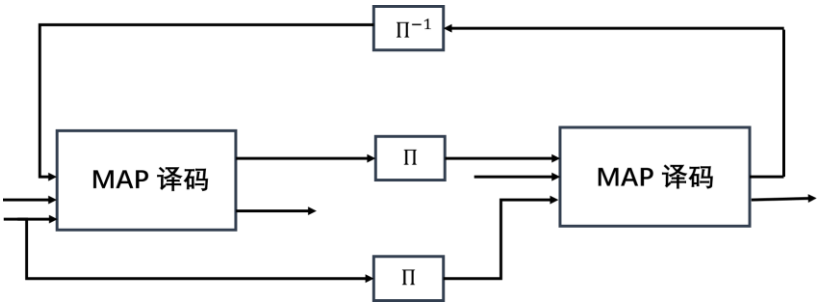


图 2.2 Turbo 译码器

经过 40 多年的研究，人们一直尝试使用信道编码来实现容量，直到 1993 年发明了 Turbo 码[20]，它使用了图 2.2 所示的迭代译码算法。Turbo 码的工作原理是让两个译码器相互产生所谓的先验概率，并且对于每一次迭代，正确判决的概率都会变得更大。

Turbo 码是编码理论的重大突破，几年后，LDPC 码[21]在 60 年代发明后被重新发现。LDPC 码工作原理是，它有一个非常稀疏的编码矩阵，而且这种稀疏性很容易用一个图来表示。在译码器上，译码器迭代地交换图中的先验概率，以使判决更加具有鲁棒性和纠正错误的能力。这使得 LDPC 码具有高度的并行性，适用于高速率的应用程序。

虽然可以通过经验证明已经接近了信道容量，但仍然不能证明上述任何一种编码能够在任意信道上实现超高容量。此外，尽管这些码的性能非常接近信道容量，但信道译码仍然是任何现代芯片设计者[22]最消耗精力和领域的任务之一。因此，在信道编码的理论和实践方面仍有很大的改进空间。

Turbo 码和 LDPC 码是目前大多数通信标准所使用的领先信道码，直到几年前，这一主导地位预计将继续保持下去。但是对于新的应用来说，需要探索其他

能够满足未来通信标准要求的信道编码方案，极化码就是其中之一。

极化码，由 Erdal Arikan 在 2009 年发明，被证明可以实现信道容量，但实际应用有限。最近，在相似的计算复杂度下，它们的实际性能可以与最近的 LDPC 码和 Turbo 码相提并论，有的时候更有甚者领先于它们[23]。由于这种性能，在过去的几年中，极化码受到了广泛的关注，因此被选为第五代移动通信标准的编码方案。然而极化码不同于 Turbo 码和 LDPC 码，它不是迭代译码，尽管极化码的许多方法借鉴 LDPC 编码理论，但在许多方面有很大不同，如极化码的凿孔以及在深度学习方面的译码应用，这将是本文的重点。

2.1.2　极化码

信道极化是一种效应，即一组信道使用 W^N ，每一个具有对称容量的 $I(W)\in(0,1)$ ，被转化为一组信道 W_N^i ，其容量 $I\left(W_N^i\right)$ ，趋向于 0 或 1，即要么完全有噪声，要么完全无噪声。这种极化已经被证明适用于无记忆的信道和有记忆的信道[23]，以及二进制和非二进制的输入符号[24]，但在这里我们只讨论 Arikan 最初的二进制方案[25]。

Arikan 极化方案是递归的，首先将两个具有相同统计量的独立信道 $W(y|x)$ ，变成一个矢量信道，使用我们的信息位 u。

$W_2:x^2\to y^2$，如：

$$W_2(y_1|y_2|u_1,u_2)=W(y_1|u_1\oplus u_2)W(y_2|u_2) \quad \cdots\cdots (2.1)$$

而对于任意的 N，$W_N:x^N\to y^N$，因为：

$$W_N\left(y_1^N|x_1^N\right)=W^N\left(y_1^N|u_1^NG_N\right) \quad \cdots\cdots (2.2)$$

其中，G_N 被定义为：

$$G_N=B_NF^{\otimes n} \quad \cdots\cdots (2.3)$$

其中 $N=2^n$ ，B_N 是比特反转互换矩阵，克朗克乘积定义为：

$$F^{\otimes n}=F^1\otimes F^{\otimes n-1}=\begin{pmatrix}F^{\otimes n-1} & 0\\ F^{\otimes n-1} & F^{\otimes n-1}\end{pmatrix} \quad \cdots\cdots (2.4)$$

其中，$F^1=\begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}$。

在图 2.3 中可以看到w_2和w_4的这种转变。可以看到u_i和x_i的关系是$G_4=\begin{pmatrix}1 & 0 & 0 & 0\\ 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 0\\ 1 & 1 & 1 & 1\end{pmatrix}$。

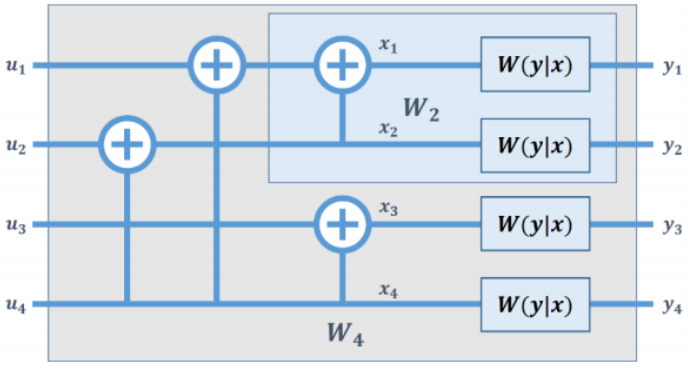


图 2.3 递归通道转换

矢量信道本身不是很有用，在论文的其余部分，将它称为比特信道，其形式为$W_N^{(i)}:x\to y$。因此，对于任意的 N，信道以下列方式被“分割”。

$$W_N^{(i)}\left(y_1^N,u_1^{i-1}\mid u_i\right)=\sum_{u_{i+1}^N\in x^{N-i\frac{1}{2N-1}}}W_N\left(y_1^N\mid u_1^N\right) \quad \cdots\cdots (2.5)$$

(2.5)的拆分操作和(2.2)的合并操作可以合并为一个单一的操作，当$N=2$时，

$$W_2^{(1)}\left(y_1^2\mid u_1\right)=\sum_{u_2\in\{0,1\}}\frac{1}{2}W\left(y_1\mid u_1\oplus u_2\right)W\left(y_2\mid u_2\right) \quad \cdots\cdots (2.6)$$

$$W_2^{(2)}\left(y_1^2,u_1\mid u_2\right)=\frac{1}{2}W\left(y_1\mid u_1\oplus u_2\right)W\left(y_2\mid u_2\right) \quad \cdots\cdots (2.7)$$

$$\begin{aligned}W_{2N}^{(2i-1)}\left(y_1^{2N},u_1^{2i-2}\mid u_{2i-1}\right)=&\sum_{u_{2i}\in\{0,1\}}\frac{1}{2}W_N^{(i)}\left(y_1^N,u_{1,odd}^{2i-2}\oplus u_{1,even}^{2i-2}\mid u_{2i-2}\right)\\ &\times W_N^{(i)}\left(y_{N+1}^{2N},u_{1,even}^{2i-2}\mid u_{2i}\right)\end{aligned} \quad \cdots\cdots (2.8)$$

$$\begin{aligned}W_{2N}^{(2i-1)}\left(y_1^{2N},u_1^{2i-2}\mid u_{2i-1}\right)=&W_N^{(i)}\left(y_1^N,u_{1,odd}^{2i-2}\oplus u_{1,even}^{2i-2}\mid u_{2i-2}\oplus u_{2i-2}\right)\\ &\times W_N^{(i)}\left(y_{N+1}^{2N},u_{1,even}^{2i-2}\mid u_{2i}\right)\end{aligned} \quad \cdots (2.9)$$

比如，对于 N=4 的信道，分割以及合并的操作将出现这样的情况：

$$W_4^{(1)}\left(y_1^4,u_1\right)=\sum_{u_2\in\{0,1\}}\frac{1}{2}W_2^{(1)}\left(y_1^2,u_1\oplus u_2\right)W_2^{(1)}\left(y_3^4\mid u_2\right) \quad \cdots\cdots(2.10)$$

$$W_4^{(2)}\left(y_1^4,u_1\mid u_2\right)=\frac{1}{2}W_2^{(1)}\left(y_1^2,u_1\oplus u_2\right)W_2^{(1)}\left(y_3^4\mid u_2\right) \quad \cdots\cdots\cdots\cdots(2.11)$$

$$W_4^{(3)}\left(y_1^4,u_1^2\mid u_3\right)=\sum_{u_4\in\{0,1\}}\frac{1}{2}W_2^{(2)}\left(y_1^2,u_1\oplus u_2\mid u_3\oplus u_4\right)W_2^{(2)}\left(y_3^4,u_2\mid u_4\right)\cdots\cdots(2.12)$$

$$W_4^{(4)}\left(y_1^4,u_1^3\mid u_4\right)=\frac{1}{2}W_2^{(2)}\left(y_1^2,u_1\oplus u_2\mid u_3\oplus u_4\right)W_2^{(2)}\left(y_3^4,u_2\mid u_4\right)\cdots\cdots(2.13)$$

在这四个方程（2.10）-（2.13）中，我们可以观察到一些重要的特性。在方程（2.10）中，$u_1=\arg\max_{u_1\in\{0,1\}}W_4^{(1)}\left(y_1^4\mid u_1\right)$ 。在随后的方程（2.11）-（2.13）中，我们依赖于之前的判决u_1，u_2，u_3 。因为这个过程是逐位处理的，所以译码可以被看作是有顺序进行的。极化是指使用方程（2.6）和（2.7）从 W 合成的信道$W_2^{(1)}$和$W_2^{(2)}$，满足以下关系：

$$I\left(W_2^{(1)}\right)\le I\left(W\right)\le I(W_2^{(2)}) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（2.14）$$

这意味着从$W\left(y\mid x\right)$ ，我们已经合成了一个较好的信道，和一个较差的信道。如果我们继续这样做，从$W_4^{(1)}$合成两个新的通道，例如$W_4^{'(1)}$和$W_4^{'(2)}$ ，然后从$W_2^{(2)}$合成两个新的通道$W_4^{''(1)}$ 和$W_4^{''(2)}$，那么我们可以看到，容量已经偏离了原来的容量$I\left(W\right)$。

$$I\left(W_4^{'(1)}\right)\le I\left(W_2^{(1)}\right)\le I\left(W\right)\le\left(W_2^{(2)}\right)\le I\left(W_4^{''(2)}\right)\cdots\cdots\cdots\cdots（2.15）$$

$$I\left(W\right)=\frac{I\left(W_2^{(1)}\right)+I\left(W_2^{(2)}\right)}{2} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（2.16）$$

而当我们以 n 个步骤递归地应用这个方法来创建 N 个比特通道$W_N^{(i)}$时容量就被保留了：

$$I\left(W\right)=\frac{\sum_{i\in E}I\left(W_N^{(i)}\right)}{N} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（2.17）$$

Arikan 表明，随着 N 的增长，当$I\left(W_N^{(i)}\right)=1$时，原始容量的比特信道的比例

将收敛到 $I\left(W_0^0\right)$；当 $I\left(W_N^{(i)}\right)\approx 0$ 时，原始容量的比特信道的比例将收敛到 $1-I\left(W_0^0\right)$；不等于 1 或 0 的比特信道的部分将收敛为 0。

2.2 极化码的编码算法

2.2.1 极化码的编码原则

信道编码在以下两种情况下是微不足道的：$I(W)=0$ 时，即不可能进行通信，没有必要采用信道编码；$I(W)=1$ 时，即信道是完美的，不需要信道编码。因此，一组 N 个极化信道的编码操作很简单。如果 $I\left(W_N^{(i)}\right)=0$ ，我们“冻结”这个比特通道，即没有信息，并设置 $u_i=0$；如果 $I\left(W_N^{(i)}\right)=1$ ，此时我们的信息 $u_i\in\{0,1\}$，完全没有编码。在译码器一侧，冻结的位置需要在编码器和译码器一侧知道，因此我们只需要关注非冻结位。

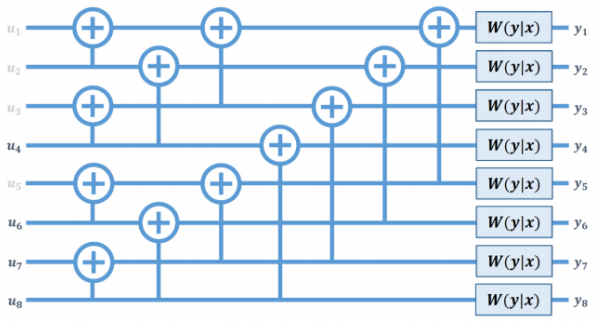


图 2.4 极化编码。u 的阴影指数被冻结，即 $u_i=0$

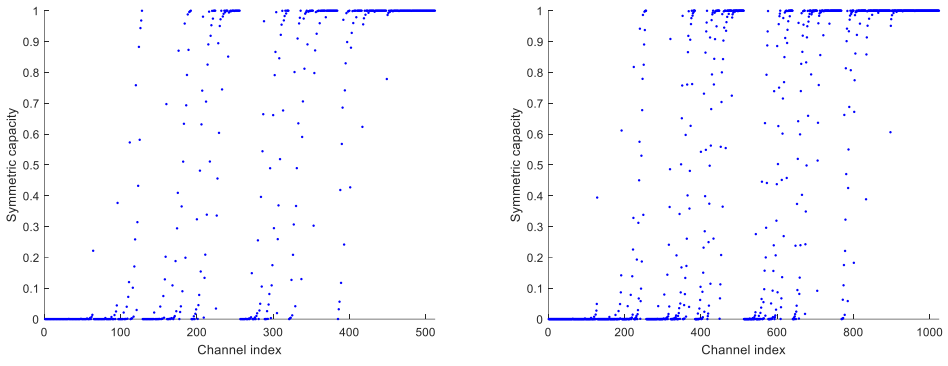
选择将信息放在哪个位置上的算法或方法通常被称为“编码构造”。其思路是找到具有最低错误概率的信道指数集 $W_N^{(i)}$，其中 $\arg\min A_C,|A_C|=N-K\sum A_C P_e\left(W_N^{(k)}\right)$ ，A_C 是具有高错误概率的信道指数集。图 2.4 中可以看到 N=8 和 R=0.5 的代码构造的例子。这里有 4 个位置被冻结，用阴影来表示。编码等于 $x=uG_8$ ，其中 $u_{1,2,3,5}=0$ 。

BEC 是一个容易分析的信道，对于极化码来说，它有一个特殊的性质，即码的构造可以精确表达。每个比特通道的可靠性 $Z(W)$，可以精确计算出来的。

$$Z\left(W_n^{(2i-1)}\right)=2Z\left(W_{n-1}^{(i)}\right)-Z\left(W_{n-1}^{(i)}\right)^2 \quad \cdots\cdots (2.18)$$

$$Z\left(W_n^{2i}\right)=Z\left(W_{n-1}^{(i)}\right)^2 \quad \cdots\cdots (2.19)$$

其中递归从 $Z\left(W_0^0\right)=Z(W)=\varepsilon$ 开始，其中 a 是擦除概率，$N=2^n$ 是代码的块长。在图 2.5 中显示了一些具有不同擦除概率的 BEC 的递归，图 2.5(a)为码长 N=512,图 2.5(b)为码长 N=1024。可以看出，随着递归步骤 n 的增加，分布变得更加极端。



(a) N=512　　(b) N=1024

图 2.5 巴氏参数的分布

我们将好的信道集表示为 A，是通过取 K 个最低值 $Z\left(W_n^{(i)}\right)$。正如我们之前提到的，Bhattacharyya 参数是单次使用信道 W 的错误概率的上限。在（2.18）和（2.19）中，Bhattacharyya 参数将是位信道 $W_N^{(i)}:x\to y$ 的错误概率的上限。

通过将位通道 $W_N^{(i)}:x\to y$ 的每个错误事件看作是独立于其他错误事件的 $W_N^{(i)}:x\to y$，对于相同用途的通道 W_N，我们可以通过对 Bhattacharyya 参数的求和得出块错误概率的上界。这将是一个上界，因为通道 $W_N^{(i)}$ 和 $W_N^{(j)}$ 的错误事件很可能是相互依赖的，因为如果我们在译码中出现错误，这个错误很可能会由于连续相消译码器而传播。使用这些 Bhattacharyya 参数，我们可以得到一个块错误率的上限为：

$$P_{upper}\left(\varepsilon\right)=\sum_{i\in A}Z\left(W_N^{(i)}\right) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2.20)$$

2.2.2　基于密度进化的编码算法

虽然 Bhattacharyya 参数在计算 BEC 信道中虚子信道可靠性方面简便易行，但实际通信环境中，很多信道类型为 BSC 或 BAWGN。由于 Bhattacharyya 参数方法中的式（2.15）只在 BEC 信道情况下适用，其在其他信道可靠性估计上存在局限。因此，Mori R 和 Tanaka T 在 T. Richardson 和 R. Urbanke 的密度演化估计

Bhattacharyya 参数的基础上，提出了一种密度演化法来衡定极化信号通道的可行性。这类应用模式一般应用在所有 B-DMC 信号通道。对于密度演化作用发展在极化码信号通道选择上的研究文献相对较少，本章对此进行了深入探讨。

信号通道极化可经过应用 LDPC 码的 Tanner 译码树结构来表示反映。以 N=8 为案例，如下示意设计图 2.6 所示：图里加粗的途径表示反映第 i 个比特（i=4）的译码途径。已经了解比特位的途径在译码过程里用虚线表示反映（由于极化码运用逐比特译码，第 i 个比特的译码策略和以前比特的译码结果相关）。不关系第 i 个比特译码的比特位用细线表示反映。

圆形和矩形符号分别表示改变量控制节点和校对核验控制节点。改变量控制节点保存该比特取值是 0 或者 1 的几率，而校对核验控制节点则应用在运算和临近改变量控制节点比特数值校对核验和是 0 的实际状况。为精简处理运算，改变量控制节点中一般保存对数似然比（LLR，也就是运算公式（2.21））。针对所有比特的相关计算，按照图 3-2 中的顺序，从右向左进行。

$$L_N^{(t)}\left(y_1^N,\hat{u}_1^{t-1}\right)=\ln\left(\frac{W_N^{(t)}\left(y_1^N,\hat{u}_1^{t-1}|0\right)}{W_N^{(t)}\left(y_1^N,\hat{u}_1^{t-1}|1\right)}\right)\cdots\cdots\cdots\cdots\cdots\cdots\cdots\text{（2.21）}$$

密度演化方法在评估虚子信道的可靠性时，将 Tanner 译码树中某一改变量控制节点的 LLR 看作是随机改变量。设 $a(x)$ 表示其概率有效密度数学函数（PDF），对于对称的 B-DMC 信道，当发送全零比特时，相应变量节点的判断错误概率可表示为式（2.22）：

$$P_e=\int_{-\infty}^{\infty}a(x)dx\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\text{（2.22）}$$

以 Tanner 译码树为根本基础，从最左端的第 i 个改变量控制节点开始出发，朝右侧拓展至最右端产生一棵树。设 a_W 表示最右端改变量控制节点的概率有效密度，也就是 $a_1^{(1)}$；用 $a_N^{(i)}$ 表示反映第 i 个比特 $L_N^{(i)}\left(y_1^N,u_1^{t-1}\right)$ 的概率有效密度。在输送了全零比特并且前 i-1 个比特 u_1^{t-1} 已知时，基于式（2.21）与（2.22）的密度演化理论，第 i 个虚子信道的 LLR 值的概率密度函数（PDF）可以通过以下递归方法得到，具体可参见式（2.23）、式（2.24）和式（2.25）：

$$a_{2N}^{(2t-1)}=a_N^{(2)}\odot a_N^{(t)}\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\text{（2.23）}$$

$$a_{2N}^{(2t)} = a_N^{(i)} * a_N^{(t)} \quad (2.24)$$

$$a_1^{(1)} = a_u \quad (2.25)$$

其中，“⊙”和“*”分别表示校验节点和变量节点的卷积操作。“⊙”为 LLR 域展开域转换的分布卷积计算，而“*”为普通分布卷积。在获取 $a_N^{(i)}$ 之后，可以通过（2.22）计算第 i 个虚子信道的误码率，从而估算可靠性。密度演化原理可以通过式（2.23）、式（2.24）和式（2.25）得出第 i 个比特的 PDF。关键是“⊙”和“*”的递归计算。本次深入研究分析重点针对 AWGN 信号通道，所以对 AWGN 信号通道的两大类计算展开具体说明。

高斯信道的信道模型可以通过图 2.7 简易表示反映。应用 BPSK 调节控制，用+1 和-1 依次表示反映 0 和 1。“*”计算是 LLR 域（简单称之为 L 域）的普通分布卷积，“⊙”是 G 域的圆圈分布卷积，其基本计算模式同时也是经过普通分布卷积展开协同计算的 4。要运算第 i 个信号通道的概率有效密度，要求在两大类分布卷积计算时，展开一个 L 域到 G 域转换。假设 A 代表 L 域变量 z 的分布，a 是其 PDF；A 代表 G 域变量 z 的分布，a 是其 PDF。两个域之间的分布变换遵循表 2.1。

表 2.1 L 域和 G 域分布的关系

L to G	G to L
$A(1, \mathrm{z} \ge 0) = 1 - \mathrm{A}^{-}(\ln\coth(\mathrm{z}/2))$	$\mathrm{A}(\mathrm{z} \ge 0) = 1 - A^{-}(1, \ln\coth(\mathrm{z}/2))$
$A(-1, \mathrm{z} \ge 0) = \mathrm{A}(-\ln\coth(\mathrm{z}/2))$	$\mathrm{A}(\mathrm{z} \le 0) = A(-1, \ln\coth(-\mathrm{z}/2))$
$a(1, \mathrm{z} \ge 0) = \mathrm{a}(\ln\coth(\mathrm{z}/2))/\sinh(\mathrm{z})$	$\mathrm{a}(\mathrm{z} \ge 0) = a(1, \ln\coth(\mathrm{z}/2))/\sinh(\mathrm{z})$
$a(-1, \mathrm{z} \ge 0) = \mathrm{a}(-\ln\coth(\mathrm{z}/2))/\sinh(\mathrm{z})$	$\mathrm{a}(\mathrm{z} < 0) = a(-1, \ln\coth(-\mathrm{z}/2))/\sinh(\mathrm{z})$

在 G 域中，“⊙”操作是针对 $F_2 \times [0, +\infty)$ 的执行。对于两个分布 $1_{\{s=1\}}\mathrm{A}(1, \mathrm{x}) + 1_{\{s=-1\}}\mathrm{A}(-1, \mathrm{x})$ 和 $1_{\{s=1\}}\mathrm{B}(1, \mathrm{x}) + 1_{\{s=-1\}}\mathrm{B}(-1, \mathrm{x})$，进行圆圈卷积时，计算规则遵循式（2.26），从而获得圆圈卷积后的 G 域分布。此处，“*”代表 L 域的常规卷积运算。

$$\begin{aligned} A \odot B = & 1_{\{s=1\}}(A(1,\cdot) * B(1,\cdot) + A(-1,\cdot)B(-1,\cdot)) \\ & + 1_{\{s=-1\}}(A(-1,\cdot) * B(1,\cdot) + A(1,\cdot) * B(-1,\cdot)) \end{aligned} \quad (2.26)$$

据表 2.1 可知，在 L 域中，自主改变量的取值有效分布范围为$(-\infty,+\infty)$，在 G 域，自主改变量的取值有效分布范围为$[0,+\infty)$。式（2.26）中的$1_{\{s=1\}}$和$1_{\{s=-1\}}$依次表示反映从 L 域到 G 域的转换过程里，L 域正半部分与负半部分的转换。设 BAWGN 信道的概率密度遵循 N~$\left(2/\sigma^2,4/\sigma^2\right)$，其对应的 L 域 PDF 应为:

$$a_{\mathrm{BAWGN}}(z)=\sqrt{\frac{\sigma^2}{8\pi}}e^{\frac{\left(z-\frac{2}{\sigma^2}\right)^2\sigma^2}{8}} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (2.27)$$

对于常规分布卷积和圆圈分布卷积的递归运算，要求在 L 域和 G 域相互之间展开转换。但是，这类转换根据非线性关联数学函数 y=lncoth（x），这给卷积计算带来了一定程度的困难。在域变换之后需要进行处理，再根据式（2.50）进行常规卷积，才能执行相关计算。按照式（2.23）、（2.24）及（2.25），可根据奇数索引和偶数索引分别进行递归计算。密度发展进化提高法经过估测子信号通道误码率P_e来评测子信号通道的可行性，并且实现信号通道选择。它一般应用在任何种类的 B-DMC 信号通道。但是，在真实完成过程里，因为要求$a_N^{(t)}$保存、具体标准量化等操控管理，一般需保护一个高分布维度的分布矢量，一般为10^6更有甚者更高。除此之外，分布卷积计算关系非线性关联计算。然而，从递归的 PDF 计算式来看，由于在由第$\log N$级获得$\log(2N)$级的 PDF 的时候，相互之间不会受到外界干扰，存在并排运算提高速度的可能性。因此，密度进化法仍具有一定的研究价值，有待进一步深入研究。

2.2.3 基于信道探测的编码算法

在处理多种通信系统的信道选择时，单一信道可靠性评价技术表现不佳。因此，研究一种适用于不同通信信道的子信道可靠性评估方法具有显著价值。参考文献[26]中指出了一类信号通道检测技术，该技术经过比特信号通道熵系数来判定信号通道的安全可靠程度。当比特信道熵参数较高时，虚子信道的不确定性增大，从而导致传输过程中出现信息差错的风险增加，适宜用于传输双方已经了解的固定比特。但是，当比特信号通道熵系数偏低的时候，自动输送过程里的随机性降低，数据信息出现错误的概率降低，因此适用于传输信息比特。

值得注意的是，这一方法以信道探测为基础，因此其适用范围较广。同时，

该方法具备实时信道探测能力，使其在应对信道突然或频繁变化的情况下具有实际应用意义。信道探测过程通过使用发送和接收端都已经知的数据通讯消息，例如全零比特序列，在通过极化程序代码调节控制后展开自动输送，同时在自动接收控制终端实行 SC 译码。在多次探测过程统计后，计算信道熵的平均值。在 N 个信号通道熵中选择较低的 K 个子信号通道应用在自动输送数据信息比特，而较大的 N-K 个子信号通道，应用在自动输送双方已经了解的固定比特。

研究证实，此模式具备较高的鲁棒性，虽然在较大的极化码码长作用范围里，某一信噪比下所选用的数据信息信号通道，一般应用在一定作用影响范围的信噪比信号通道选择，而不会导致显著的误码率性能损失。

2.3　极化码的译码算法

在本章中，我们将为极化码实现实用的译码算法。第 2.2 节中提到的原始极化码的问题之一是，尽管极地码使用逐次消除译码器可以证明随着 N 的增长而达到容量，但与 Turbo 码和 LDPC 码相比，中等块长的性能并不是很好。因此，我们将首先实现原始的译码，然后提出并分析能显著提高中等块长性能的译码算法。

我们将解释、实现和评估一组不同译码器的性能。我们将专注于基于 LLR 的译码器，因为与基于 LR 和名义似然的译码器相比，这类译码器要简单得多，并且避免了其他译码器中存在的一些数字问题。我们还假设我们有冻结集和非冻结集，分别是 A 和 Ac 。

2.3.1　极化码的译码原则

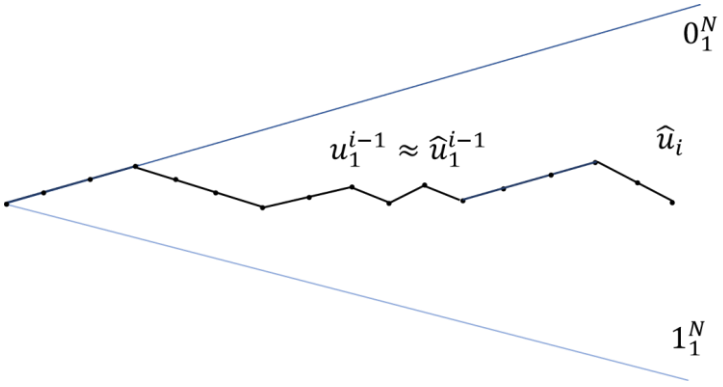


图 2.6 连续相消译码下的译码路径

Arikan 利用逐次相消译码制定了极化编码方案。连续相消译码是一种贪婪的搜索算法，它使用精灵辅助算法依次对每个比特做出决定。

$$u_i = \underset{u_i \in \{0,1\}}{\arg\max} WN\left(y_1^N, u_1^{i-1} \mid u_i\right) \quad \cdots\cdots\cdots\cdots\cdots\cdots (2.28)$$

在这个意义上，它需要正确的决定 u_1^{i-1} ，这是参数辅助的。这些通常对译码器来说是不可用的，所以它可以通过设置 $u_1^{i-1} = u_1^{i-1}$ ，如图 2.6 所示的近似。然而，这使得它容易出现错误传播，因为当出现错误时，该错误会影响到之后的判决。

2.3.2 基于 LLR 的译码算法

对于 SC 译码器，基于 LLR 的译码的定义如下。

对于 $i \in A_c$:

$$u_i = 0 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2.29)$$

对于 $i \in A$ ：

$$u_i = \begin{cases} 0, \ln\left(\dfrac{w\left(y_1^N, u_1^{i-1} \mid 0\right)}{w\left(y_1^N, u_1^{i-1} \mid 1\right)}\right) > 0 \\ 1, otherwise \end{cases} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2.30)$$

意思是说，如果索引位置被冻结，那么我们将译码输出设置为零，如果索引位置没有被冻结，那么我们将使用（2.30）中的 LLR 来决定输出。

由于极化码的构造完全是递归的，我们将通过推导出 $N = 2^n$ 的译码操作来展示其原理。

Y_{LR} 我们将似然比定义为 $Y_{LR} = \dfrac{W\left(y_1^N, u_1^{i-1} \mid 0\right)}{W\left(y_1^N, u_1^{i-1} \mid 1\right)}$ ，并使用递归公式（2.6）和（2.7）推导出 LLR，表示为 Li。

$$L_1 = \ln\left(\frac{w_2^{(1)}\left(y_1^2 \mid 0\right)}{w_2^{(1)}\left(y_1^2 \mid 1\right)}\right) = \ln\left(\frac{\gamma_{y_1} * \gamma_{y_2} + 1}{\gamma_{y_1} + \gamma_{y_2}}\right) \quad \cdots\cdots\cdots\cdots\cdots\cdots (2.31)$$

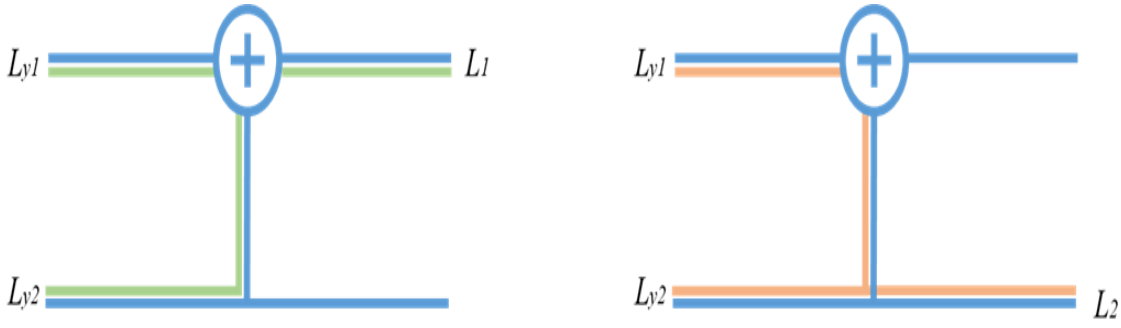
$$L_2=\ln\left(\frac{w_2^{(2)}\left(y_1^2,u_1\mid u_2=0\right)}{w_2^{(2)}\left(y_1^2,u_1\mid u_2=1\right)}\right)=\begin{cases}\ln\left(\gamma_{y_2}*\gamma_{y_1}\right),u_1=0\\ \ln\left(\dfrac{\gamma_{y_2}}{\gamma_{y_1}}\right),u_1=1\end{cases} \quad \cdots\cdots(2.32)$$

如果 L_{y_1} 和 L_{y_2} 是来自信道的接收 LLR，那么基于 LLR 的更新方程如下，我们使用了适当的近似值。

$$L_1=\operatorname{sgn}\left(L_{y_1}\right)\operatorname{sgn}\left(L_{y_2}\right)\min\left(\min\left|L_{y_1}\right|,\left|L_{y_2}\right|\right) \quad \cdots\cdots(2.33)$$

$$L_2=\begin{cases}L_{y_2}+L_{y_1},u_1=0\\ L_{y_2}-L_{y_1},u_1=1\end{cases} \quad \cdots\cdots(2.34)$$

可以看出，这里的方程非常简单，由和符号和比较运算符组成，非常适用于硬件实现。这使得基于 LLR 的译码与使用名义分布值 $W(y|0),W(y|1)$ 或 LR 函数相比更加稳健。式(2.33）和（2.34）有时被称为图 2.7 中的 f 函数和 g 函数。



（a）f 函数　　（b）g 函数

图 2.7 基本译码元素

2.3.3　基于连续相消的译码算法

对于 SC（连续相消）译码器，译码过程是按顺序一步步进行的。译码器必须在译码过程中做出判决，然后在译码过程中使用这些判决。在冻结位置的情况下，无论 LLR 的值如何，译码器都会将判决设置为零。除了 LLR 计算（2.33）和（2.34）之外，图中的判决还必须通过以下公式在图中传播。

$$u_{2i-1}(k-1)=u_{2i-1}(k)+u_{2i}(k) \quad \cdots\cdots(2.35)$$

$$u_{2i}(k-1)=u_{2i}(k) \quad \cdots\cdots(2.36)$$

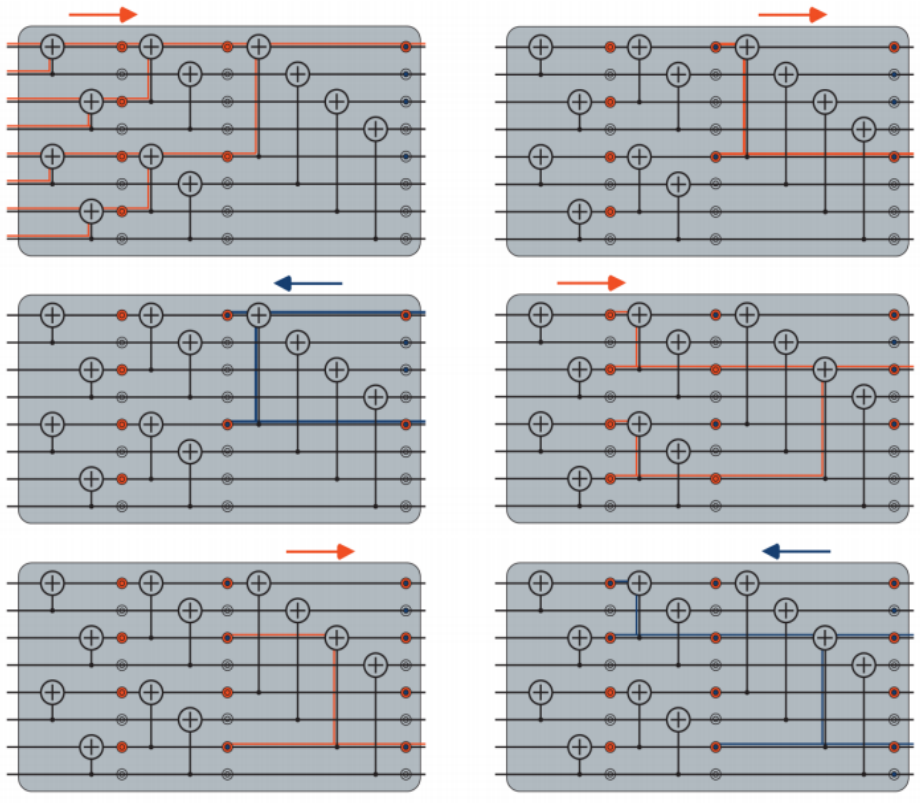


图 2.8 $N=8$ 的连续相消译码的六个第一步

$N=8$ 极化码的译码图和译码的六个首要步骤见图 2.8。在图 2.8 中 a) 译码器首先通过 f 函数（2.33）从最低层到第一个决策-LRs，但指数已经冻结。c) 决定 a)和 b)通过(2.34)和(2.35)传播。 d) 这里我们先用 g 函数计算 LLR，然后用 f 函数计算。

下面介绍译码的算法。这种译码算法大体上遵循[17]。由于译码是递归的，所以将有两个递归函数，递归计算 LLR 和递归传播比特。递归计算 LLR 从左到右计算 LLR，递归传播比特算法 2.1 中第 11 至 15 行译码器做出的决定。

Algorithm 2.1: Successive Cancellation Decoder

Input : the received vector **y**, with size ***N***

Output:approximated information sequence ***u***

1 for $\beta=0, 1, \ldots N-1$

2 $L(\beta,0)=4\frac{y_\beta}{N_0}$

续表

Algorithm 2.1: Successive Cancellation Decoder	
3	for ϕ=0, 1, …, N-1
4	*recursivelyCalculateLLR* (m, ϕ)
5	if $\boldsymbol{u}$ is frozen
6	$B_m(\varphi,0)=0$
7	$u(\phi)$=0
8	else
9	if $L(0,\ \phi)>0$
10	$B_m(\phi,0)=0$
11	u(ϕ)=0
12	else
13	$B(\phi, 0)$=1
14	$u(\phi)$= 1
15	if ϕ *mod*2=1
16	recursivelyPropagate$B(\phi, m)$
17	Return: $u(\phi)_N^2=\phi$

在算法 2.1 中，前两行将通道 LLR 加载到变量 L(β, λ)上，该变量在图中保存 LLR。然后，循环遍历所有的比特，其中第 6-8 行在冻结时将输出设为零。第 10-15 行是决策函数，进行估计决策 u_i 和 $B_m(\phi,0)$ ，其中 $B_m(\phi,0)$ 是保存比特决策的变量。

Algorithm 2.2:recursively Calculate ***LLR(λ, ϕ)***	
Input:layer $\boldsymbol{\lambda}$ and phase $\boldsymbol{\phi}$	
Output:log likelihood ratio ***LLR(λ, ϕ)***	
1	ψ=[ϕ/2]

续表

Algorithm 2.2:recursively Calculate LLR(λ, ϕ)	
2	if λ=0
3	return
4	if ϕ *mod* 2=1
5	recursively Calculate LLR(λ-1, ψ)
6	for β=0, 1, …, $2^{m-\lambda}$
7	if ϕ mod 2
8	$L(\beta,\lambda)=f\left[L(2*\beta,\lambda-1),L(2*\beta-1,\lambda-1)\right]$
9	else
10	$L(\beta,\lambda)=g\left[L(2*\beta,\lambda-1),L(2*\beta-1,\lambda-1)\right]$

在算法 2.2 中，通过 f 和 g 函数在每个 j 上递归计算 LLR。而在算法 2.3 中，式(2.30)和(2.31)的传播是在奇数索引位置处进行的。

Algorithm 2.3:recursively Propagate ***LLR(λ, ϕ)***	
Input : layer ***λ*** and phase ***ϕ***	
Output:log likelihood ratio ***LLR(λ, ϕ)***	
1	Ψ=[ϕ/2」
2	for $\beta=0,1,\ldots,2^{m-\lambda}$
3	$B(2*\beta)=B(\beta,0)\oplus B(\beta,1)$
4	$B(2*\beta+1,1)=B(\beta,1)$
5	if ψ *mod* 2= 1
6	recursively Propagate LLR(λ, ϕ)

2.3.4 基于连续相消列表的译码算法

SC 译码器在性能方面的主要问题是，一旦做出错误的比特决定，就没有办法纠正这个错误的决定，因此肯定会出现块错误。为了使逐次相消译码器不犯这

些错误，在[17]中提出了连续相消列表（SCL）译码器，它极大地改善了性能。然而，值得注意的是，Reed Muller 码[18]的列表译码器，其结构与极化码[19]非常相似，早已为人所知。

列表译码器使用与逐次相消译码器相同的译码操作，但在算法 2.4 的第 13 行，译码器不是只做一个决定，而是做两个决定，将译码路径分成两个译码路径，在下一个索引位置，它将把这两个路径分成四个路径。然后继续这样做，直到由于路径太多，不得不将这些路径去除掉。这种去除是基于给每个路径的路径度量。这个路径度量是对该路径是正确信息序列 u 的可能性的近似值。低度量表示好的路径，高度量表示坏的路径。在长度为 $N=8$ 的译码中，显示了从索引位置 2 到 7 的六个译码步骤。

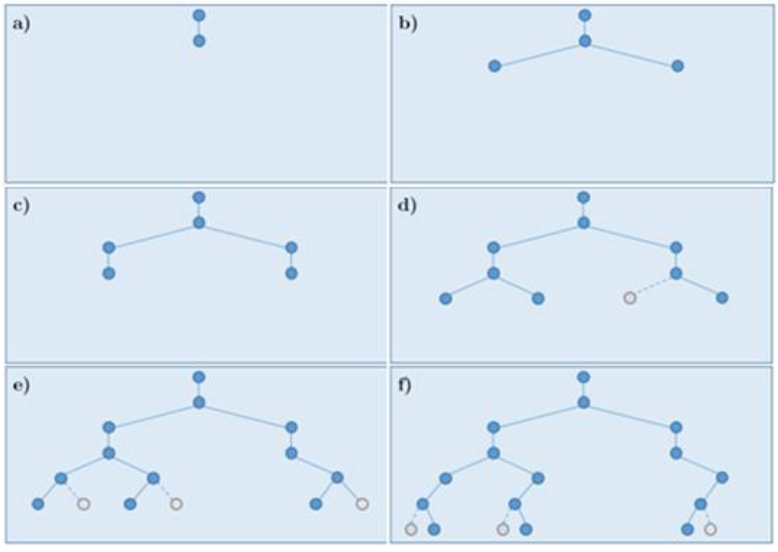


图 2.9 SCL 译码器的路径进展

灰色的节点表示已关闭的路径，蓝色表示开放的路径。在图 2.11 中，a)图中迭代在 j=1 处。 b)图中第一个解冻的索引位置，被分成两条路径，$u_2(l=0)=0$ 和 $u_2(l=1)=1$。 c)图中这个索引被冻结，因此除了 $u_3(0)=0$ 和 $u_3(1)=0$，没有任何判决。 d)图中我们把两个路径分成四个路径，但是我们的列表大小是 $L=3$，因此其中一个分割路径不会被延长。e)图中我们再次拆分路径，然后根据拆分后的路径度量决定再次使用哪条路径。f)图中在每个解冻的索引上重复同样的程序，但是如果没有一条路径被扩展，那么直到根节点的那条路径将被干掉。

这个译码算法的复杂度是 $O(LN\log(N))$，但是它也需要大量的内存和排序操作来对每个非冻结索引的路径进行排序。下面算法 2.4 是 SCL 译码算法。

Algorithm 2.4:Successive Cancellation List Decoder

Input : the received vector ***y***, with size ***N***

Output : approximated information sequence ***u***

1	$l_1 = assignInitialPath$
2	for β=0, 1, …N -1
3	$L_{l_1}(\beta,0) = 4\frac{y_\beta}{N_0}$
4	for ϕ=0, 1, …, N-1
5	recursively Calculate List LLR(m, ϕ)
6	for l=0, 1, ..., L-1
7	if u_ϕ is frozen
8	if path l is active
9	$B_m(\phi,0,l) = 0$
10	$PM_l = CalcPathMetric\left(L_\varphi^l, PM_l, 0\right)$
11	$u(\phi,l) = 0$
12	else
13	path Pruning(ϕ)
14	if i *mod* 2=1
15	*recursively Update List* $B(\phi,m)$
16	$l_{min} = \underset{l\in L}{arg\ min}\ PM_l$
17	Return: $u(\phi,l_{min})_N^1 = \phi$

列表产生后，如第 18 行所示，选择具有最佳路径度量的路径作为输出信息

序列。图 2.11 a)-d)显示了该译码器在不同块长和列表大小 L 下的性能。在算法 2.5 中，显示了路径修剪的操作。这是 SCL 译码器最重要的操作之一。在第 1 行到第 9 行，所有的活动路径 *Pactive* 被扩展，并计算出它们的路径度量。在第 10 行到第 16 行，我们分析并决定在这些扩展的路径 *Pextended* 中，哪些应该继续下去，哪些应该被除掉。这是通过根据路径的度量值进行排序，并选择 L 个最好的 *Pextended* 来完成的。如果两条扩展路径都没有被选为 L 条幸存的路径之一，那么我们将杀死根路径 *Pactive*。

Algorithm 2.5: *path Pruning*

1	for l=0, 1, …, L-1
2	if path l is active
3	*path Mettric Forks*(l, 0)=*CalcPath Metric*($L_{\phi}^{1}, PM_{l}, 0$)
4	*path Mettric Forks*(l, 1)=*Calc Path Metric*($L_{\phi}^{1}, PM_{l}, 1$)
5	i=i+1
6	else
7	path Metric Forks(l, 0)=∞
8	path Metric Forks(l, 1)=∞
9	σ= min(2i, L)
10	*continue Probability*=sort *path Metric Forks* and set L last indices to 0, and L first indices to 1
11	for l=0, 1, ..., L-1
12	if path l is active

续表

Algorithm 2.5: *path Pruning*	
13	if both *continue Probability* of l is zero
14	kill Path(l)
15	for l=0, 1, ..., L-1
16	if both continue Probabilities(l)=1
17	$B_m\left(0,\phi \bmod 2,l\right)=0$
18	$u\left(\phi,l\right)=0$
19	$PM_l = CalcPathMetric(L_{\phi}^{l}, PM_l, 0)$
20	$l_{clone}=clonePath\left(l\right)$
21	$B_m\left(0,\varphi mod\,2,l_{clone}\right)=1$
22	$u\left(\varphi,l_{clone}\right)=1$
23	$PM_{l_{clone}}=CalcPathMetric\left(L_{\varphi}^{l_{clone}},PM_{l_{clone}},0\right)$
24	else
25	if continue Paths(l)=1
26	$B_m\left(0,\phi mod\,2,l\right)=0$
27	$u\left(\phi,l\right)=0$
28	else
29	$B_m\left(0,\phi mod\,2,l\right)=1$
30	$u\left(\phi,l\right)=1$

我们使用与[16]中描述的相同类型的路径管理。路径管理需要的一组重要操

作是杀死和复制/开辟新的路径。对于基于 LLR 的实现，这是通过让每条路径用一个路径度量来表示。算法 2.4 第 11 行的路径度量被更新如下：

$$PM_{\varphi}^{l}\left(L_{\varphi}^{l},PM_{\varphi-1}^{l},u_{i}^{l}\right)=PM_{\varphi-1}^{l}+\ln\left(1+\exp\left(-\left(1-2u_{\varphi}^{l}\right)L_{\varphi}^{l}\right)\right)\cdots\cdots\cdots(2.36)$$

其中，l 是路径编号，j 是比特索引。

2.3.5　基于 CRC 辅助的连续相消列表的译码算法

与 SC 译码器相比，SCL 译码器的性能要好得多，但与 LDPC 码和 Turbo 码相比，其性能仍然不具竞争力。人们注意到，在许多错误案例中，正确的码字在最后的输出列表中，但由于相应的路径度量不是最小的，所以没有被选中[17]。我们的想法是实现一个“精灵”，它能够在列表中找到正确的码字。这个“精灵”可以通过附加一个 CRC 来实现，然后选择 1）通过 CRC 检查和 2）具有最佳路径度量的码字。该方案如图 2.10 所示，在第 18-20 行修改后的 CA-SCL 算法如算法 4.6 所示。CRC 被附加在错误概率最低的索引上，以确保 CRC 得到很好的保护。

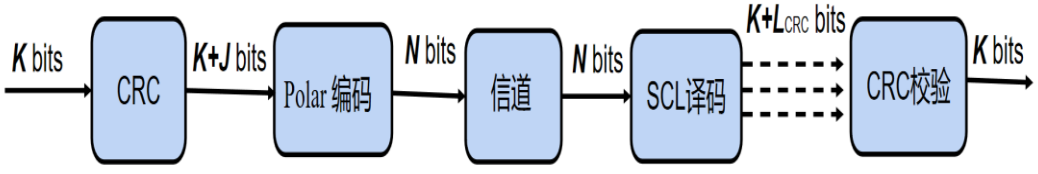


图 2.10 CRC 辅助的 SCL 译码器方案

Algorithm 2.6:CRC-aided Successive Cancellation List Decoder

Input:the received vector ***y***, with size ***N***

Output:approximated information sequence ***u***

1　　$l_1 = assignInitialPath$

2　　for β =0, 1, …N-1

3　　$L_{l_1}(\beta,0)=4\dfrac{y_\beta}{N_0}$