密级：　　保密期限：

北京郵電大學

硕士研究生学位论文

题目：基于数字喷泉码的研究及其应用

学　　号：076304

姓　　名：周　阳

专　　业：信号与信息处理

导　　师：张　琳

学　　院：信息与通信工程

2010 年　3 月　12 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

基于数字喷泉码的研究及其应用

摘　要

数字喷泉码是一种新型的删除编码，它可以产生无限的输出符号，并灵活地进行码率控制。在实现上，它采用了一种单向的异步传输机制，具有高效、低延迟的性能特征，并对信道的时变性有很高的适应能力。此外，由于其编译码复杂度较低，因此也非常容易实现。

由于其种种优势，数字喷泉码具有非常广泛的应用场景，它也逐步进入到实用领域中。迄今为止，已有不少相关的技术专利面世；同时，一些国际组织也将其纳入到相关标准中，包括 3GPP、DVB、IETF 等，其发展前景非常可观。

在本文中，我们介绍了数字喷泉码的发展历程，并针对 LT 码、Raptor 码等几种典型的实现方案进行了详细的说明。通过整理与归纳，我们总结了该领域目前的问题以及研究现状。

针对 LT 编码中出现的重复关联问题，我们从理论上对它做了详细的分析与讨论，揭示了产生无效编码符号的原因，并推导出无效符号的发生概率。进一步地，我们设计出几种可行的改进方案，通过对随机关联进行限制来有效地防止重复关联。

针对分布式应用场景，我们就 Raptor 码进行了实现与讨论，并结合该场景与编译码的特点进行分析，阐明了度 1 符号的重要性，以及 Parity 在分布式场景中的特殊性。进一步地，我们提出了一种基于

度 1 符号的改进设想，并设计出具体的编码优化算法。

关键字　　数字喷泉码　无码率　LT 码　Raptor 码　分布式信源编码

THE RESEARCH AND APPLICATIONS BASED ON DIGITAL FOUNTAIN CODES

ABSTRACT

Digital fountain codes (DFC) are a new class of erasure codes. They can yield limitless encoding symbols and have flexible rate-control. In implementation, DFC adopts a simplex mechanism of asynchronous transfer, which possesses high-efficiency and low-delay, and adapts the time-variant of channel easily. In addition, because of its low complexity of coding, DFC is easy to realize.

Due to the aforementioned advantages, DFC can be applied in wide scenarios, and it enters into practical field gradually. So far, many patents are proposed, and DFC is accepted by some international standards, such as 3GPP, DVB, IETF, etc. Therefore, DFC is very promising in the future.

In this paper, the history of DFC is referred, and we introduce some typical schemes in detail, including LT codes and Raptor codes. By some inductions, the related problems and the current researches are concluded.

For the problem of duplicate associates in practical LT encoding, we conduct some detailed analyses and discussions theoretically. The reason why invalid encoding symbols are produced is revealed, and the

probability that invalid encoding symbols appear is inferred. Moreover, we propose some improved algorithms which can eliminate duplicate associates effectively by the restriction of random associations.

For distributed applications, the scheme of Raptor codes based is realized and discussed. By combining the characteristics of the scenario and coding, we conduct some analyses. And the importance of degree-one symbols (DOS) and the specificity of parity in distributed scenario are illustrated. Furthermore, we propose an assumption of DOS, and design an optimized encoding algorithm.

KEY WORDS digital fountain codes, rateless, LT codes, Raptor codes, distributed source coding

目录

符号对照表

3GPP/3GPP2：3rd Generation Partnership Project (2)，第三代通信规范合作计划（2）

ALC：Asynchronous Layered Coding，异步分层编码

AWGN：Additive White Gaussian Noise，加性白高斯噪声（信道）

BER：Bit Error Rate，误比特率

B(M)SC：Binary (Memoryless) Symmetric Channel，二元（无记忆）对称信道

BP：Belief Propagation，置信传播（算法）

CDN：Content Delivery Network，内容分发网络

CS：Check Symbol，校验符号

DFC：Digital Fountain Codes，数字喷泉码

DISCUS：Distributed Source Coding Using Syndromes，基于 Syndrome 的分布式信源编码

DOS：Degree-One Symbol，度 1 符号

DSC：Distributed Source Coding，分布式信源编码

DVB(-H)：Digital Video Broadcasting (-Handheld)，（手持式）数字视频广播

DVC：Distributed Video Coding，分布式视频编码

FEC：Forward Error Correction，前向纠错

FLUTE：File Delivery over Unidirectional Transport，单向传输中的文件发送（协议）

GCD：Greatest Common Divisor，最大公约数

IETF：Internet Engineering Task Force，互联网工程任务小组

IP：Internet Protocol，互联网协议

IS：Input Symbol，输入符号

ISD：Ideal Soliton Distribution，理想的孤波分布

LDPC：Low Density Parity Check，低密度校验（码）

LLR：Log-Likelihood Ratio，对数似然比

MBMS：Multimedia Broadcast Multicast Service，多媒体广播组播服务

MP：Message Passing，信息传递（算法）

LRLTC：Limited Randomness LT Codes，随机性有限的 LT 编码

OS：Output Symbol，输出符号

PDA：Personal Digital Assistant，个人数码助理（掌上电脑）

RSD：Robust Soliton Distribution，健壮的孤波分布

TCP：Transmission Control Protocol，传输控制协议

UEP：Unequal Error Protection，不等差错保护

WSN：Wireless Sensor Networks，无线传感器网络

第一章 绪论

1.1 研究背景和意义

上个世纪出现了计算机与互联网，并在一批批杰出的技术和科研人员的不懈努力下，使其成为了一项成熟而且便捷的生活用具。尤其是后者，它不仅为我们的信息传递和交换提供了快速的方式；更重要的是，它超越了实际的物理距离，拉近了人们之间交流的通道，使我们的整个世界俨然成为了一个“地球村”。可以说，IT 和互联网时代带给了人们前所未有的新体验。

当今社会，我们随处可以看见计算机的影子，而且上网也不再像以前那么奢侈了。现代科技的飞速发展，使人们的生活水平也有了显著提高，这其中包括我们的日常起居、交通、娱乐等各个方面。此外，人们在享受丰富物质文化生活的同时，也对互联网有了更高的要求：如何提高传输速率，保证更好的传输质量；是否能够提供移动性的上网接入方式（比如手机）……诸如此类，这一系列的需求和问题，也进一步地推动了互联网技术的发展。

1.1.1 互联网上的数据传输

互联网是一个庞大的数据传输体系，它采用分组（包）交换的方式，在网络 TCP/IP 协议栈的严格控制下进行着数据传输。为了有效地完成差错控制，协议栈对网络架构进行了分层设计，每一层根据相应的控制信息（包头、校验位等）来完成指定的功能，并将其内容部分往上（往下）层交付。

与物理层采用的比特级纠错技术不同，上层协议处理的对象通常是一个个分组（包）单元。在实际的数据通信中，为了保证传输的可靠性，发送端通常在原始分组中加入一些校验信息，这使得分组内部具有健壮的检验能力，这样接收端可以通过这种机制来检验该分组是否存在误码并予以丢弃。可见，这种传输特点表现出明显的删除信道（Erasure Channel）特征：一个包要么被正确收到；要么由于误码、拥塞、错误路由等原因而被丢弃。如图 1-1 所示，这里，p 表示删除概率（即错误概率），X 表示被丢弃的错误符号。

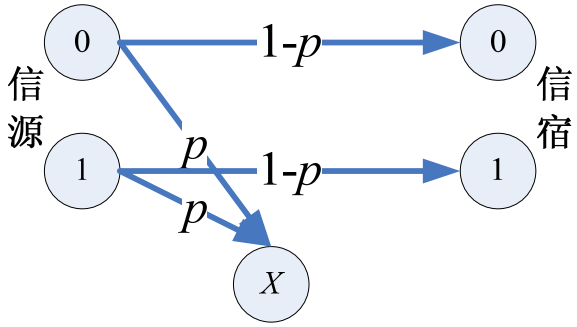

图 1-1 删除信道示意图

传统的删除编码通常是码率固定的块码（Block Codes）[1]，即 K 个输入符号对应 N 个输出符号，码率固定为 K/N。比较典型的是 Reed-Solomon 码，其特点是：任意接收到的 K 个符号都能恢复出原始信息来。不过，如果信道质量太差，往往会引起大量的重传，致使降低信道效率；另外，它的编译码复杂度很高，不适合分组数目较大的情况。还有一种是 Tornado 码，它是一类 LDPC 码，其特点是：线性的编译码复杂度，而译码只需略大于 K 个符号就能完成。虽然 Tornado 码在复杂度上有比较可观的优势，但仍然无法摆脱传统删除编码的致命弱点，其编码设计都要基于一定的先验信道假设。而在实际应用中，由于信道的时变性，很难保证稳定的信道条件，倘若考虑根据信道条件不断更新码字构造，这又会使复杂度大大增加。因此，传统的删除编码无法实现完美的可靠传输。

此外，网络协议本身也存在一定的不足，比较典型的问题就是 TCP 中的反馈重传机制，它需要一个反馈信道（在资源有限的情况下，这本身也是一种奢侈的要求），通过发送、应答的方式同步地完成传输。这也造成了一些传输隐患：

- 当信道质量很差时，不断地重传反馈会引起较高的时延，从而降低传输效率，这是极为浪费的开销；
- 对于长距离数据传输，由于每次要保证收发的同步性，这也势必造成很高的延迟；
- 在组播/广播的应用场景中，这显然是不合适的，如果发送端每次都要接收大量的反馈信息，并可能为一小部分用户重新发送所有数据，这会造成许多冗余，也会造成较高的时延。

由于传统方案中存在的种种不足，我们亟需寻求一种新的差错控制技术，以进一步改善当前网络的传输模式。

1.1.2　新一代差错控制技术——数字喷泉码

在前面一系列问题的襁褓中，数字喷泉码（Digital Fountain Codes, DFC）应运而生。喷泉码，顾名思义，就是像喷泉（编码器）一样，可以不断涌出水珠（输出信息），其中每一滴都具有相等的信息价值；因此，接收端只需用水杯（译码器）装满足够数目的水珠就能满足要求（译码成功），如图 1-2。由于数字喷泉码可以产生无限的输出符号、灵活地码率控制，所以又叫做无码率编码（Rateless Codes）。另外，其译码端只需很低的接收开销，仅 $K(1+\varepsilon)$个编码符号（ε 是一个极小的正数）就能完成译码，并且它只看中数目，而不“关心”某些特定符号的接收，这使其具有非常好的灵活性与可扩展性。

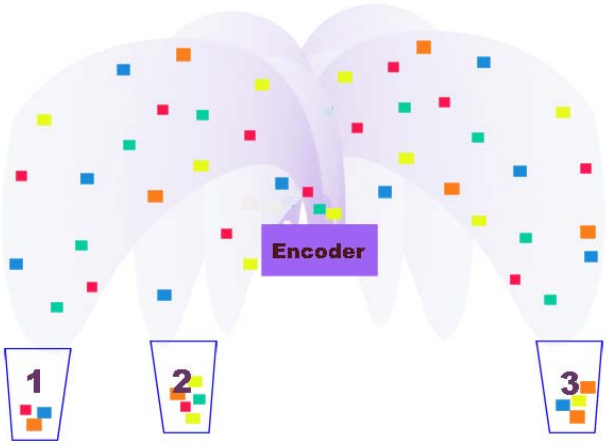

图 1-2　数字喷泉码示意图

数字喷泉码是一种前向纠错（FEC）技术，它无需反馈信道，并解决了传统删除编码中码率固定的问题。同时，也不用先验假设的信道条件，并能够适应信道的时变性。此外，其编译码复杂度较低，也易于实现。另一方面，数字喷泉码弥补了传统协议栈的不足，它采用了一种单向的异步传输机制，具有高效、低延迟的体系框架，并且无需收发中的同步响应，还能应用于组播/广播的场景中。由于其实现简单，该技术可以在低功率的平台上应用，尤其是一些用户级移动电子产品，比如手机、PDA 等，其性能优势更是显著。可以说，数字喷泉码既保证了传输的可靠性，同时又满足了操作上的有效性。

1.1.3　数字喷泉码的应用现状

数字喷泉码最大的特点在于其灵活的码率控制，因此可以适用于几种典型的应用场景[2]：单发单收、单发多收、多发单收、多发多收，如图 1-3 所示。

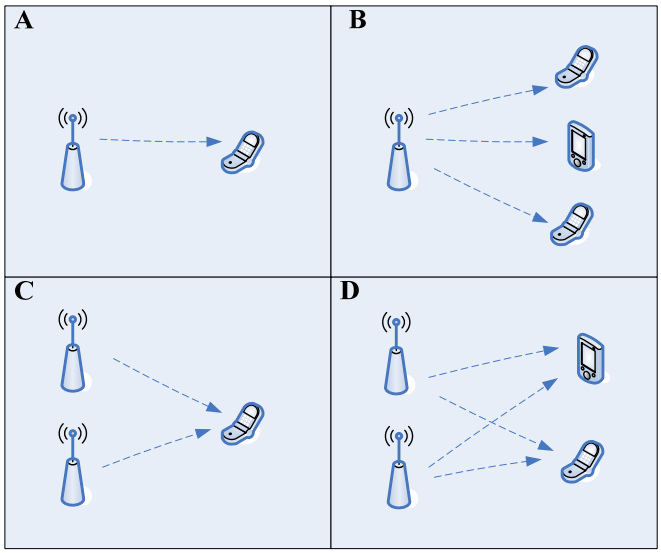

图 1-3　四种场景

A-单发单收、B-单发多收、C-多发单收、D-多发多收

这里，需要说明的是：

- 在多发情况下，由于数字喷泉码码字构造的随机性和低重复性，使得多个发送端产生的编码冗余度非常低，而且相互之间无需事先进行同步“协商”，甚至不用同时启动（随机接入即可）；因此，发送端越多，其接收端译码效率越高；
- 在多收情况下，由于数字喷泉码各码字之间具有统计上均等的信息量，所以，译码的成功与否并不依赖于某些特定码字的接收情况；换句话说，只要收到足够数目的码字，即可完成译码操作，同时，由于其弹性的速率控制，达到这个目的是容易实现的。

以上面四种基本类型为依托，数字喷泉码技术可扩展到一些更具体的实用领域。目前为止，该技术已经应用于一些商用产品[3]：

- IPTV，针对不同的媒体类型、编码格式、压缩、数据率和加密等处理，DFC 可以轻松完成兼容操作，并提供高质量、低开销的视频效果；
- 移动广播，DFC 可以无视数据率、丢包率、断断续续连接状况的影响，而只需低成本的网络基站部署，便可实现点到点、点到多点的数据传输；此外，在系统架构上，只需在传输层以上架设一个虚拟的子层，以完成差错控制环节；与传统方案相比，其速度更快、范围更广、内容更丰富，且不易受网络条件的限制；
- 内容分发网络（CDN），DFC 可以实现真正意义上的端到端传输，它可以改善一般网络视频的慢载入、频繁缓冲、低画质的不足；通过完善的策略管理系统，还能解决“最后一公里”问题；

- 军事防御系统，DFC 可以提供精确、实时的数据传输，并能对抗战场中恶劣的物理环境（信道时变性），可在带宽、延迟、衰落、中断等多种限制条件下，实现可靠传输。

与此同时，相关的知识产权、技术专利等也先后发布于世。目前，典型的 LT 码、Raptor 码以及系统码的实现方式已经发布[4、5、6]。在文献[7]中提出了一种基于分组的实现方案，可以更灵活地控制数据源。考虑到 Raptor 码的多信道传输特性，文献[8]中给出了一种具有纠错能力的实现方案。

此外，由于数字喷泉码的种种优势，它也逐步受到国际社会的关注，一些国际组织已将其纳入了相关标准之中[9]：

- IETF，DFC 是 FLUTE/ALC/FEC 单向文件传输构架的主要贡献者；它还完成了 RMT 小组有关文件传输标准的最后评论要求；此外，DFC 还被列入了其 FECFRAME 计划之中；
- DVB，其中的 DVB-H 接受了 FLUTE/ALC/FEC 架构；另外，R10 FEC（系统 Raptor 码）还用到了其文件传输服务中；
- 3GPP，其文件传输标准接受了 FLUTE/ALC/FEC 架构；同时，R10 FEC 还用到了 MBMS 服务中；
- 3GPP2，文件传输标准接受了 ALC/FEC 架构，并正在考虑是否接受 FLUTE 和 R10 FEC 技术。

另一方面，受到数字喷泉码优越性能的吸引，一些新领域的研究者也逐步把目光投入到其中，这里比较典型的就是在分布式场景和认知网络场景中的应用。

在分布式场景中，考虑到多信源间的相关性，可以对其进行联合压缩。而进一步的分布式编码理论[10]提到：信源端无论是否进行联合编码，在译码端通过联合译码都能够实现相同的译码效果。因此，可以对信源进行相对简单的独立编码，从而简化信源端的构造复杂度，达到低功耗的需求。不过相应地，译码端的负担也会随之而增加，需要设计更为复杂的算法来进行联合译码。所以，分布式编码并没有完全简化整个编码体系，而是将信源端的编码复杂度转移到了译码端，降低了编码的开销而增加了译码的难度。这是一种折中，它正好适合于一些需要低功耗传输的特定场景，比如无线传感器网络（WSN）。分布式编码在实现上的一个关键技术就是使用一种信道码（纠错码，例如 LDPC 码），通过只传输编码后的冗余信息达到独立压缩的效果。而数字喷泉码正好是一种性能优越的信道码，因此将其应用与分布式场景自然是一种可行的方案。

认知网络[1]是一个新兴的领域，其目的在于提高无线系统的资源利用率，增加资源分配上的灵活性。这里，在网络架构中有一个重要的设计思路：由于主用户通常是少数，并且其频带使用率较低，所以，可以考虑将主用户的频带资源在

适当的时候分配给多数的次级用户使用，从而提高系统的资源利用率。当然，这里还有一个重要的前提，即这种分配上的灵活转换必须保证主用户在资源使用上的无障碍优先性。此外，还应尽可能地减小切换中的资源开销，不然就可能得不偿失了。

由于数字喷泉码具有灵活的码率控制，因此，它可以很好地解决切换中的开销问题。如图 1-4 所示，在 A 图中，初始情况下，主用户 1、主用户 2 分别空出了各自的资源 R2 分配给次级用户使用；在 B 图中，主用户 2 收回了 R2，次级用户只剩主用户 1 的 R2 可用，实际操作时，主用户 2 直接抢占使用资源即可，无需其他任何多余的开销，而对于次级用户来说，它对特定的数据并没有要求，就像图 1-2 中的杯子一样，它只关心实际的“水量”，因此，主用户 2 的这种抢占只是影响了次级用户的传输速率，导致译码时间延长，但并不影响最终的译码性能，也不会造成多余的切换开销；在 C 图中，主用户 3 又空出了 R1、R3 供次级用户使用，此时，新加入的资源直接通过数字喷泉码技术进行编码传输即可，无需顾及其他资源（例如先前已经使用主用户 1 的 R2）的传输情况，而对于次级用户来说，这也只是加速了传输速率，也无需任何切换处理。

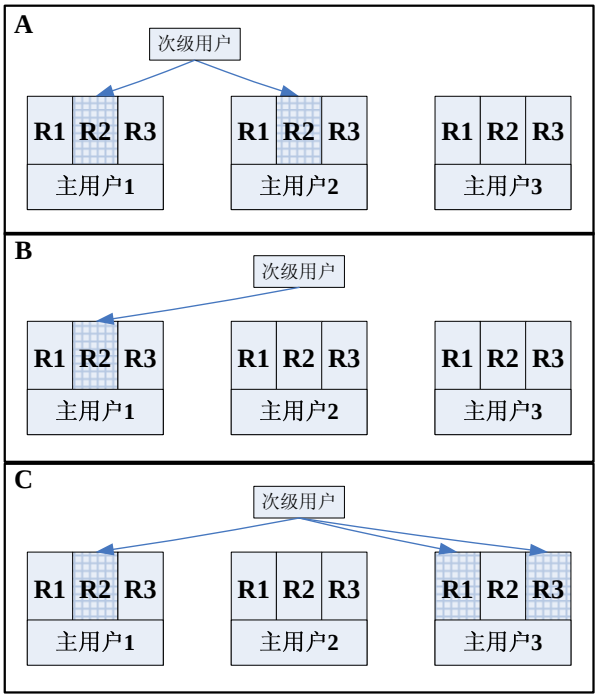

图 1-4　数字喷泉码在资源切换中的使用示例

A-初始情况、B-主用户收回资源、C-主用户提供资源

综上所述，数字喷泉码已逐步成为一项成熟且实用的差错控制技术，在无数学者和技术人员的不断努力下，其发展前景必将无限光明。

1.2 本文的着眼点以及结构安排

数字喷泉码的优势十分明显，它不仅具有比较完善的理论体系，而且也已进入到实用化阶段，其研究价值不言而喻。在本文中，我们从基本的数字喷泉码入手，先后讨论了其理论依据和实现方案，并分析了当前存在的问题和研究现状。接着，针对实现上产生的重复关联问题进行了讨论、分析，并提出了改进策略。另一方面，基于分布式应用场景，我们采用数字喷泉码进行了仿真实现，并验证了其可行性。

下面的内容结构安排如下：

- 第二章中，我们将先对数字喷泉码的历史及其理论进行讨论。然后，针对最典型的两种数字喷泉码方案——LT 码、Raptor 码进行详细的说明，并揭示其优越性的本质。最后，介绍了当前在理论和技术上还存在的一些问题以及研究现状。
- 第三章中，我们先从实现入手，阐明了编码的随机性所造成的重复关联问题。然后，通过分析和推理，找到了该问题产生的原因；同时，文中也对其出现的概率进行了严格的数学推导，证实了它的严重程度。接着，我们设计出几种改进的编码方案，通过一定的限制条件，杜绝了重复关联。最后，利用仿真实验，验证了改进方案的正确性。
- 第四章中，我们针对当下比较热门的分布式编码场景，使用 Raptor 码进行了实现和一些性能仿真。接着，通过分析，我们结合分布式系统的特点与 Raptor 译码算法的特殊性，提出了一种基于度 1 符号的优化设想，并在不影响原码字性能的前提下，设计出具体的编码算法。最后，通过仿真验证了该方案的合理性与可行性。
- 第五章中，我们回顾了全文的内容，并强调了本文研究的几个重点问题和研究成果。进一步地，基于目前的研究情况，对未来更深入的研究方向进行了展望。

第二章 数字喷泉码理论概述

数字喷泉码的概念最早是由 M. Luby 于 1998 年首次提出的，但当时仅限于描述性介绍，并没有给出具体的设计方案。2002 年，受到稀疏二部图的启发，M. Luby 在此基础上提出了世界上第一种可行的数字喷泉码，并命名为 LT 码（Luby Transform Codes）[11]。LT 码是一种线性信道码，其编码操作简单，具有灵活的码率控制，并能实时地产生编码符号。此后，A. Shokrollahi 在 LT 码的基础上提出具有线性复杂度的 Raptor 码[12]，其核心思想在于级联了一个外部的线性编码器，通过适当地增加冗余来降低译码复杂度。由于 LT 码和 Raptor 码都是非系统码，而实际应用中，我们往往更希望采用系统码的方式，这样可以简化译码操作，更快捷地获取原始信息。于是，A. Shokrollahi 进一步给出了一种设计系统 Raptor 码的方案[12]，该方案在编码中进行了适当的矩阵变换，保持了原有非系统 Raptor 码的特点，同时具有极佳的性能指标。

下面的部分，我们将依次介绍两种典型的数字喷泉码，其中包括它们的编译码算法、理论分析等。之后，我们还会列举出数字喷泉码目前还存在的问题和研究现状。

2.1 LT码

LT 码是最早的数字喷泉码实现方案，也是其后出现的各种方案的鼻祖，它贯穿了数字喷泉码无码率特性的核心思想。因此，LT 码是极具代表性的一种实现方案。

在 LT 编码时[13]，编码器会先将原始信息进行分割，得到 K 个等长的输入符号 IS（Input Symbol）。然后，将这些 IS 作为一个集合，并让它与另一个输出符号 OS（Output Symbol，也是编码符号）的集合相互映射形成一个二部图，如图 2-1 所示。这里，两个集合间的具体关联是随机产生的（通常还需要一个度分布函数），而每个具体 OS 的值由其关联 IS 的值一起来确定。从图中可以看出，每个 OS 的产生是相互独立的，它所对应的关联 IS 也与其它 OS 无关。因此，OS 集合的大小可以是任意的（无码率特性），根据需要随时变化即可。最后，把生成的 OS 连同它们对应的关联信息进行打包，这样就可以在网络上传输了。

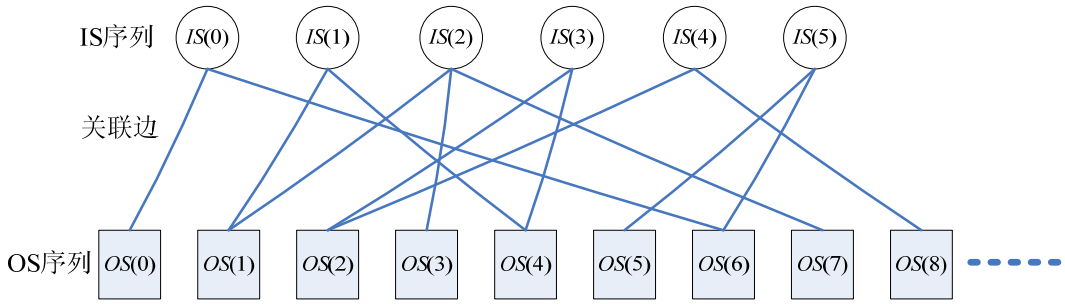

图 2-1　LT 编码示意图

译码时，译码器根据接收到的 OS 序列及其对应的关联信息，逐个完成 IS 的解析过程。由于关联方式的随机性和 OS 产生的独立性，使得每个 OS 具有均匀等值的信息负载。因此，译码的成功与否，可以无视数据损失的具体方式，而完全取决于正确接收到的 OS 数目。同时，从图中可以看出，接收端也不用针对某个特定的 OS 而煞费苦心地设计算法。此外，最终译码成功所需的 OS 数目通常只需略大于 K 即可，所以 LT 译码也是一种非常实惠的开销。

2.1.1　LT编码算法

LT 编码的具体实现算法如图 2-2：

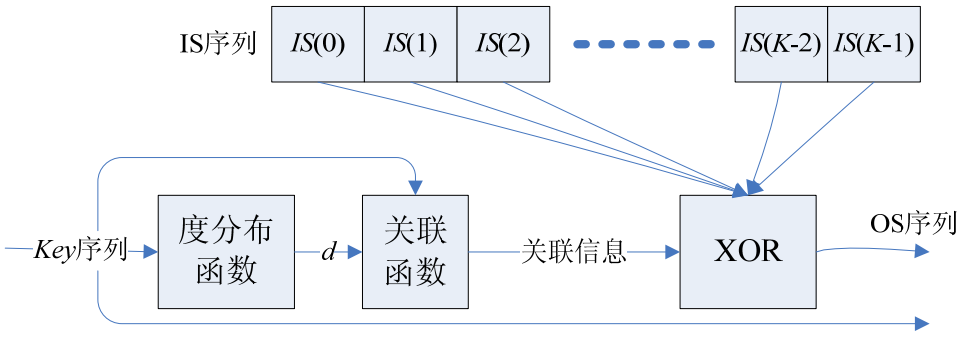

图 2-2　LT 编码算法

i.　首先，对信源数据进行分割，产生 K 个等长的 IS；

ii.　设计一个随机的度分布（Degree Distribution）函数，可按一定的概率为每个 OS 产生度数 d，这里 $d \le K$；

iii.　在 K 个 IS 中为对应 OS 均匀随机地选出 d 个关联 IS，并将它们进行异或运算，得到 OS 的值；

iv.　最后，将 OS 的值及其相关信息（度数、关联 IS）进行发送即可。

在实际处理中，为了进一步压缩关联信息，编码器会为每个 OS 预先分配一

个随机且唯一的整数 *Key*。编码时，每次由这个 *Key* 作为输入，来确定 OS 相应的 *d* 和关联 IS。这样，每次只需发送一个 *Key* 作为额外开销，就能在译码端完整地恢复出对应的关联信息了。

从 LT 编码的实现中可以看出，该方案的一个关键点在于度分布函数的设计，它必须保证所有的 IS 都能关联到 OS 中，这也直接影响最终的译码性能。M. Luby 设计了一种健壮的孤波分布（Robust Soliton Distribution，RSD）[11]，它的主要分布特点是：随着 d 的增大，其对应的概率逐步减小；但在某些个别的度数位置上可能会出现反常的高概率；还有一点，并非所有可能的度数（$1 \sim K$）都有概率，有些则可能被屏蔽了。可以证明，RSD 度分布具有非常优越的性能，它能保证译码以失败告终的概率非常小。因此，LT 编码具有很高的可靠性。

2.1.2 LT译码算法

LT 码的译码过程被称为信息传递（Message Passing，MP）[14]，它是一种迭代处理算法，其步骤大致如下：

i. 针对一个度 1（$d = 1$）的 OS，通过它可以直接推导出其关联 IS 的值；

ii. 然后，把解析出的 IS 从与之关联的其他 OS 中消去（异或运算），并更新关联信息；

iii. 接着，之前那些 $d = 2$ 的 OS 可能通过第 ii 步的消去操作变为 $d = 1$，从而回到第 i 步，进一步去触发新的迭代。

译码过程如此循环往复，形成一个“链式反应”，直到译码成功或度 1 的 OS 用尽为止。图 2-3 是一个简单的译码示例：这里，译码从 *OS*(0)开始，它可以直接推出 *IS*(0) = 1（见 B 图）；进一步地，*OS*(1)、*OS*(3)消去了 *IS*(0)，并产生新的度 1 符号 *OS*(3)（见 C 图）；继续进行下去，最终可以成功完成译码。

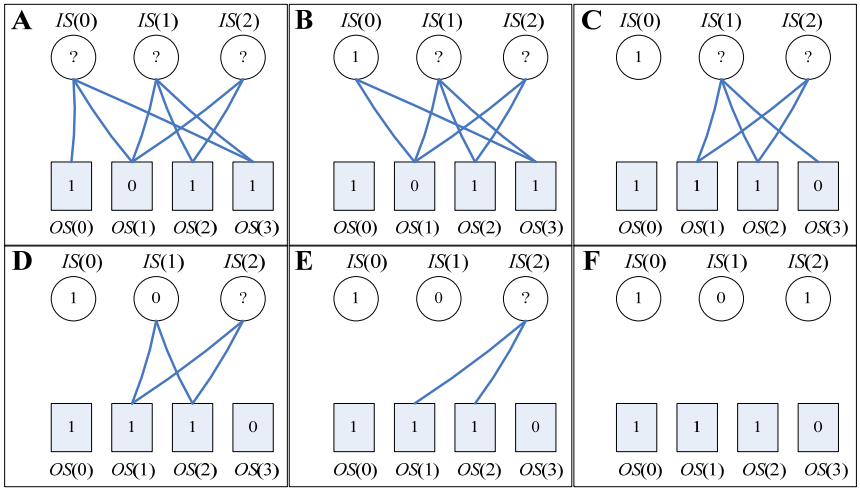

图 2-3　LT 译码过程示例[14]

从译码过程中可以看出，度 1 的 OS 扮演着非常关键的角色，每次迭代都需要从它开始，因此，它直接影响了最终的译码效果。前面提到的健壮孤波分布 RSD 度分布的优点就在于：它可以保证在译码进行中，度 1 的 OS 数目在译码成功前用尽的概率为一个任意小的正数 δ。

此外，在实际应用中，还可以对译码算法做些优化[4]：例如，每次等接收到的 OS 数目大于 K 之后再进行译码，因为从信息论的角度来说，少于 K 的 OS 根本没有足够的信息量，因此没有必要进行译码；另外，还可以考虑对 OS 按度数进行升序排列，直接依次完成度数上从小到大的译码过程，从而简化了解析关联信息中的部分开销。

综上可见，数字喷泉码性能优势的关键在于编译码算法和度分布的设计上。前者保证实际操作的可行性与有效性，而后者则要保证信息分布的均等性和译码可靠性。此外，由于其编码中 OS 产生的相互独立性，导致它具有非常灵活的码率控制能力。这就是为什么数字喷泉码可以适用于诸如多发多收这种复杂的应用场景，并且对于突发性中断、信道的时变性特征拥有独到的处理机制。

2.2 Raptor码

2.2.1 普通Raptor码

Raptor 码是以 LT 码为原型扩展而来的一种性能更优越的编码方案。由于 LT 运算复杂度为 $O(K\ln(K/\delta))$，当 K 较大时，性能会有所下降。于是，A. Shokrollahi 在此基础上提出了具有线性复杂度 $O(K\ln(1/\varepsilon))$的 Raptor 码[13]。

从设计上说，Raptor 码是一种级联码，它是在原有 LT 编码的外层“封装”了一个线性预编码（例如 Hamming 码、LDPC 码等），通过这种方式引入适当的冗余信息，从而减轻原 LT 码中过高的算法复杂度。具体地说，如图 2-4 所示，信源符号先进行了线性预编码，并得到中间阶段的 IS 集合；然后，以此 IS 集合作为 LT 码的输入，产生最终输出的 OS 序列。接收端的译码过程与之相对应：先进行内层的 LT 译码，然后是外层的线性译码，得到最终的复原信息。

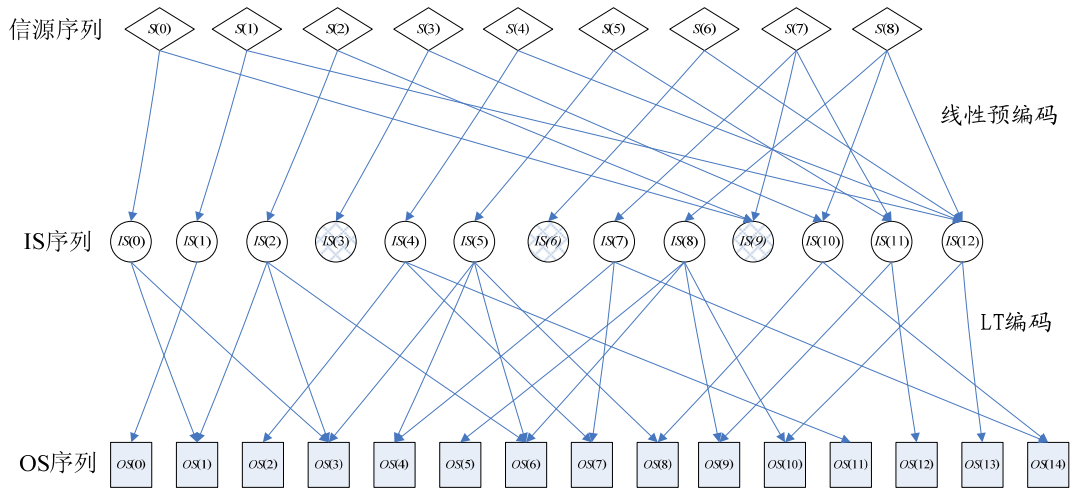

图 2-4　Raptor 编码示意图[1]

这里，由于线性预编码中引入了冗余符号，从信息论的角度，对于译码端来说，根本不必接收过多的 OS 来译出所有的 IS，而只需译出适当数目的 IS 即可。就图 2-4 而言，即使 LT 译码时无法恢复出 *IS*(3)、*IS*(6)、*IS*(9)，译码器最终也能够完整地恢复出所有的信源符号，因为这三个 IS 所关联的信源符号 *S*(0)、*S*(2)、*S*(3)、*S*(6)、*S*(7)全都已经被其它的 IS 所覆盖，因此缺少这三个 IS，不会影响最终的结果。

可以看出，Raptor 码实质上是通过适当增加线性预编码中的冗余信息，来减少 LT 译码中所须译出的 IS 数目，进而降低了 LT 译码的运算复杂度。此外，后期线性译码的算法复杂度也不高，因此，从整体上说，该方案有效地利用了预编码操作，降低了实际的编译码复杂度。

2.2.2　系统Raptor码

LT 码和普通的 Raptor 码都是非系统的编码，而实际应用中，系统码的方式往往更具有实用价值，它可以简化译码操作，更快捷地获取原始信息。于是，A. Shokrollahi 进一步从工程上给出了一种系统 Raptor 码的生成方法。该方案在普通 Raptor 码的基础上，通过一些线性的矩阵变换，实现了系统符号的集中。需要注意的是，该方案不仅产生了系统符号，而且还保持了原有非系统 Raptor 码的性能优势，因此，它也是目前数字喷泉码的主流应用技术。

下面，我们以 LDPC 码[15, 16]作外层线性预编码为例，来说明一下系统 Raptor 码的具体产生过程。

先令 $X^{[K\times 1]}$、$G_{LT}^{[N\times n]}$、$G_{LDPC}^{[n\times K]}$ 分别表示信源符号向量、LT 码的生成矩阵、LDPC 码的生成矩阵，这里 n 表示经线性预编码之后产生的中间 IS 符号向量的长度，

并且 $K \le n \le N$。于是，最终编码输出的符号向量 $V = G_{LT}G_{LDPC}X = G_{ALL}X$，其中 $G_{ALL}^{[N\times K]} = G_{LT}G_{LDPC}$。

为了产生系统的编码符号，从矩阵运算的角度来说，如果可以找到一个矩阵 $R^{[K\times K]}$，使得

$$G_{ALL}R = \begin{bmatrix} E^{[K\times K]} \\ \hline P^{[(N-K)\times K]} \end{bmatrix} \tag{2-1}$$

即产生一个单位矩阵，然后，令 $I = RX$（表示 X 的一种中间变换），可以得到变换后的编码符号为

$$T = G_{ALL}I = \begin{bmatrix} X \\ \hline PX \end{bmatrix} \tag{2-2}$$

从式（2-2）中可以发现，临时向量 T 包含了所有的系统符号，这样，就可以把它作为最终的 OS 进行传输。

以上就是系统 Raptor 码产生的基本思路，该方法的主要难点在于如何求出所需的 R。这里，为了产生式（2-1）中的单位矩阵，要求 R 是一个可逆矩阵，同时需要保证 G_{ALL} 中存在对应的逆矩阵 R^{-1}。在大多数情况下，G_{ALL} 中存在一个 $K\times K$ 的可逆子矩阵 R^{-1}，通过它就能计算出相应的 R。但是该可逆子矩阵 R^{-1} 往往并非就是由前 K 行组成，因此，需要预先进行矩阵的行交换，将实际组成 R^{-1} 的各行转移到前 K 行中。实际操作中是通过高斯消去法（Gaussian Elimination）[17] 进行处理的，该方法从上到下依次寻找出 K 个线性无关的行，并将它们先后转移到前面 K 行中，从而得到变换后的新 G_{ALL} 矩阵，以及 R^{-1} 和 R。需要注意的是，由于目的不同，这里的高斯消去法也与传统的方案有所区别：传统方案为了解出线性方程，通常会通过变换来得出一个上三角矩阵；而这里是为了求出只经过行交换后的新矩阵 G_{NEW}，因此，需要两个 G_{ALL}，一个通过传统高斯消去法得到上三角矩阵，而另一个与其一同进行行交换（不包括消去操作），这样才能得到前 K 行组成可逆 R^{-1} 的 G_{NEW}。

综上所述，可以归纳系统 Raptor 码产生的流程如图 2-5 所示：先计算出 G_{ALL}，并通过高斯消去得到行交换后的新 G_{NEW} 编码矩阵（其前 K 行构成 R^{-1}），以及可逆矩阵 R；然后，对于每一个信源符号向量 X，先乘以 R 得到中间状态 I，接着乘以 G_{NEW}，得到最终的输出向量 T，它的前面 K 位正好等于 X。

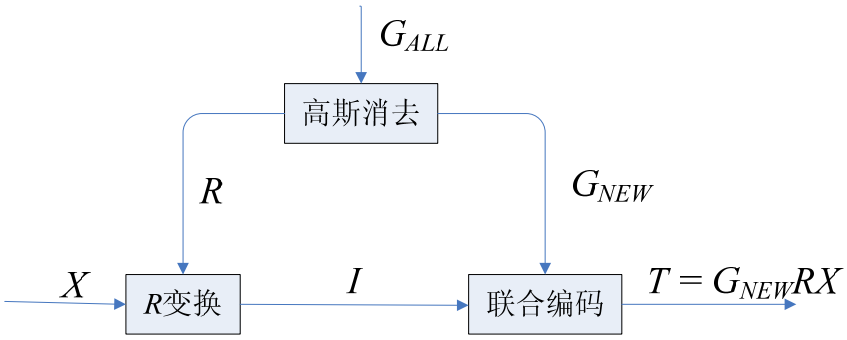

图 2-5 系统 Raptor 编码流程图

还需要说明的是，通过行交换产生新 G_{ALL} 矩阵的这种处理并不会引起原 G_{ALL} 的性能改变：如果把 G_{ALL} 分开成原来的 $G_{LT}G_{LDPC}$，那么从矩阵操作的角度上说，对 G_{ALL} 的行交换实际上只是对 G_{LT} 的行交换，这丝毫不会影响到 LDPC 码的构造方式和性能；另外，对 G_{LT} 进行行交换，对应到编码过程中就是交换两个 OS 的位置而已，从前面 2.1.1 小节对 LT 码的介绍中我们知道，OS 的产生是相互独立的，因此这种行交换也不会影响到 LT 码的性能。所以说，系统 Raptor 码不仅产生了系统符号，同时还保持了普通 Raptor 码的性能特点。

2.2.3 二元无记忆对称信道下的Raptor码

数字喷泉码是从删除信道的纠错技术发展而来的，以往所讨论的各种方案也都是针对删除信道而言的。不过，是否存在能够针对一般有噪信道的数字喷泉码方案呢？

这个答案就是 Raptor 码。由于 Raptor 码引入了线性预编码，例如 LDPC 码，它们自身就具备相当的纠错能力，因此可以很好地解决一般有噪信道的误码问题。而对于 LT 码来说，从其编译码过程来看，可以发现译码器对 OS 的正确性以来很高，因此 LT 码在一般有噪信道中显然是不适合的。

文献[18]在原有 Raptor 码的基础上做了改进，并给出了有噪的二元无记忆对称信道（Binary Memoryless Symmetric Channel，BMSC）上的 Raptor 码实现方案。在编码端，该方案在与一般的 Raptor 码一样，仍旧是先进行预编码，得到中间状态的 IS，然后利用 LT 编码输出 OS。其主要的修整是在译码器中，由于引入了有噪条件，原先使用的 MP 算法因为对高正确率的 OS 存在依赖性，因此需要调整。

在该文献中，作者使用了置信传播（Belief Propagation，BP）算法[19]，可以有效地完成 LT 译码。该算法中使用了一个对数似然比（Log-Likelihood Ratio，

LLR）的概念，表示当前符号趋于 1 或 0 的可能性，对于 BMSC 信道而言，一个符号的偏 0 型 LLR 计算公式如下：

$$R = \log(\frac{1-p}{p}) \cdot (1-2y) \tag{2-3}$$

这里，p 表示信道的差错概率，y 表示接收到的比特符号（0 或 1）。从式（2-3）可以看出，当 $p < 0.5$ 时，如果 $y = 0$，则 $R > 0$，即表示当前符号为 0 的可能性更大；如果 $y = 1$，则 $R < 0$，表示当前符号为 1 的概率较大。同理，当 $p > 0.5$ 时，也是一样的判定结果。总之，对于偏 0 型 LLR 而言，当其大于 0 时更可能为 0，否则更可能为 1。

BP 算法是一种迭代计算，它利用符号间的关联性相互传递各自的 LLR 并进行修正，这使得编码符号排除了误码干扰，最终收敛到正确的结果上。

针对 Raptor 码中的 LT 译码，其迭代算法如下：

$$\tanh(\frac{m_{o,i}^{(l)}}{2}) = \tanh(\frac{Z_o}{2}) \cdot \prod_{i' \neq i} \tanh(\frac{m_{i',o}^{(l)}}{2}) \tag{2-4}$$

$$m_{i,o}^{(l+1)} = \sum_{o' \neq o} m_{o',i}^{(l)} \tag{2-5}$$

这里，符号 o 对应 OS，符号 i 对应 IS。Z_o 表示符号 o 的 LLR，$m_{o,i}^{(l)}$（$m_{i,o}^{(l)}$）表示第 l 次迭代时，o 到 i（i 到 o）传递的 LLR。初始化时（$l = 0$），所有的 $m_{i,o} = 0$。此外，由于 tanh()函数的计算中可能涉及到无穷大的处理，因此这里还进一步定义了以下表达式：

$$\tanh(\frac{\pm\infty}{2}) = \pm 1 \tag{2-6}$$

从式（2-4）、式（2-5）可以看出，每次迭代都是将 OS 集合与 IS 集合，通过其间的关联性来进行 LLR 的传递，并在此过程中不断修正原先的 LLR。不过，需要注意的是，每轮迭代中所传递的信息，比如从 o 到 i，它是由上一轮中其它与 o 相连的 i 传递给 o 的信息组成，并不包括当前 i 传递给 o 的信息，这样能够避免 i 自身的干扰，形成良性的收敛趋势，这就是所谓的“置信传播”。

最后，迭代完毕，需要计算所有 IS 的 LLR（让 $m_{o,i}^{(l)}$ 对 o 求和），并将这些 LLR 值作为初始条件，输入到线性译码器中，并完成最终的译码。

2.3 存在的问题与研究现状

数字喷泉码发展至今，已经从纯理论的研究、实验阶段，进入到实用性、改进优化阶段。虽然该技术已经相对成熟，不过依然还存在一些理论与实践上的难

题，需要得到进一步的探索和发掘。

首先是环的问题。由于数字喷泉码中采用了随机的信息关联方式，从而可能会出现多个 IS 集中地与某几个 OS 关联的情况。从图论的角度来说，这就会形成了一个环。而从信息论的角度看，这也是一种信息冗余。为了增强编码的有效性，需要尽可能地去除这些冗余，让 IS 在 OS 中的分布进一步分散。在文献[20]中，作者设计了一种相关列提取法，以去除长度为 4 的短环，从而改善了 LT 码的性能，降低了译码复杂度。

关于输出 OS 数目的控制问题。数字喷泉码虽然是码率灵活可控的，但如何确定一个具体的 OS 数目，这也是编码端需要考虑的。极端的情况下，可以让发送端无限地发送 OS，直到接收端译码成功，并反馈一个结束信息才停止。但这种情况未免太浪费资源，而现实应用中，资源通常是受限的，因此使用无码率的特点来进行传输并不合算。这样，N 就必须使用一个有限的值。如果 N 太小，不能保证译码质量；如果 N 太大，就会造成浪费。此外，由于信道条件的时变性，更不能使用一个固定的 N 来输出。因此，寻找一种动态调整的机制来控制 N 的大小，才能比较有效地实现目标。

还有一点，当度 1 的 OS 不足时，如何完成译码呢。这里，剩余 OS 存在两种情况：有足够的译码信息、没有足够的译码信息（本来数目不足或冗余过多）。就前者而言，由于原有算法对度 1 的依赖性过高，因此需要设计一种新的译码算法解析剩余的 OS。可以考虑类似一般线性方程的解法，不过这样复杂度有点高。在文献[21]中，提出了一种“钝化”（Inactivation）的方法来处理矩阵，从而解除“死锁”。另外，文献[22]中也提出了一种增加的高斯消去法，通过新接收的 OS 来进一步扩大译码成功的概率。不过，这类方法需要根据具体的应用场景和需要来进行取舍，简单地说，如果没有其它诸如功率、实时性等限制的情况下，可以考虑使用这种高复杂度的方法进行译码。

关于不等优先级保护的问题。由于 LT 码中的随机性选择，使原始信息的关联分布上呈现均等性，即对所有 IS“一视同仁”。但是，对于一些特定的应用，某些信息可能是比较重要的，比如视频应用中的 I 帧就比 P 帧关键，如果这些信息不能恢复的话，即使其它信息都恢复出来了，也会使最终的效果变得很差。这就需要对关键信息做一些特定的保护，使其在不完全译码的时候，尽可能高地保证最终质量。在文献[23]中，提到了一种信息分级和概率重分配的方法，可以有效地保护重要信息。不过相应地，也需要牺牲复杂度作为代价。还有的文献[24]中，考虑了一种不等差错保护（UEP）的方法，也可以达到一定程度的性能提高。

此外，针对 LT 码中度分布研究也有不少。在文献[25]中指出了 RSD 度分布在实际应用中存在的不足，并设计出一种次优的度分布，通过仿真，可以证明该

方案在平均译码率、译码率方差等性能指标上优于 RSD 度分布。另一方面，文献[26]中提出了一种分布式的 LT 码，其核心思想是把信源符号分割为两部分，通过两个子编码器分别对其进行编码并发送，然后接收端可以综合两部分的 OS 进行联合的 LT 译码。这里的关键技术是也是度分布的设计，文中提到将一个 RSD 分解成两个子度分布的卷积，这样就能保证联合译码时拥有统一的 RSD 度分布，而无视 OS 具体来自哪个子编码器。

由于数字喷泉码优越的性能，它也受到一些新兴领域的青睐。例如，分布式场景，在文献[27]中，作者设计了一种用 Raptor 码实现的分布式编码方案，并提出了基于 LT 码、LDPC 码的联合迭代译码算法，最后通过仿真结果证明了其可行性。进一步地，该作者还讨论了分布式 Raptor 码在隐马尔科夫信源（Hidden Markov Sources）中的应用[28]。另一方面，认知网络也是当下研究的一个热门领域[1]，其主要思想是，把主用户空闲时期的频带资源分配给次级用户使用。但是，必须保证主用户在需要资源时的优先性。由于数字喷泉码具有灵活的码率控制，因此，它能够很好地处理资源切换中的额外开销，是一种非常有效的方案。此外，在文献[29]中，作者针对深空通信这一特定场景，设计出了基于 LDPC 码的喷泉码方案，可以有效解决该场景下高延迟、低吞吐量、终端容量和计算能力有限等问题。

2.4 本章小结

在本章中，我们先简要介绍了一下数字喷泉码的发展历程，然后针对几种典型的方案做了详细的描述。

LT 码是世界上第一种可行的数字喷泉码方案，它将灵活控制码率的思想落实到应用中。从 LT 编译码的过程来看，其关键在于度分布的设计上，为了保证较高的译码成功率，并尽可能地降低开销，M. Luby 设计出了有效的 RSD 度分布，它使 LT 码具备了更好的实用性。

Raptor 码是以 LT 码为基础扩展而来的一种级联码，它通过外加一个线性预编码，降低了运算复杂度，是一种更有效的技术。进一步地，A. Shokrollahi 在普通 Raptor 码的基础上，设计了系统的 Raptor 码，使其不仅具备了系统符号在操作上的简便性，并且保持了原有方案的性能优势。另一方面，由于 Raptor 码中预编码自身的特性，使得 Raptor 码还具有在 BMSC 信道下的纠错能力，这也使其应用范围进一步扩大了。

最后，我们讨论了数字喷泉码的研究现状，归纳了当前存在的几个难题，包括环的问题、N 的控制、译码“死锁”、优先级保护，以及度分布优化等。另一

方面，受到数字喷泉码种种优势的吸引，一些新的领域也开始尝试对其进行探索，这主要包括分布式编码方案、认知网络，以及深空通信等应用场景。

第三章 数字喷泉码中去除重复关联的方案

数字喷泉码发展至今，其理论基础已经相对完善，并且也先后公布了一系列技术专利，成为一种实用性方案，例如 LT 码、Raptor 码、系统 Raptor 码等各种技术专利，它们都已进入实施阶段。

不过，值得注意的是，当理论付诸于实现时，往往很难 100%地满足所需的外在条件。这主要是因为理论中所要求的条件通常都是比较理想化的情况，一方面它可以简化影响因素便于建立模型；另一方面也是为了从逻辑上推导出较为严格的公式结果。但在实践中，这些理想化的要求往往是很难达到的，它们都受到实际中复杂环境的变化所影响。

为了解决这些非理想因素造成的问题，实际的考虑往往是基于一种近似的处理方案，通过一些逼近的手段，把可能的变化与误差缩小到可控的范围内，从而可以得到相对合理的结果。例如，计算机中的函数计算，全部都是通过数值计算的方法来完成的，即所有函数解析式都是采用多项式级数展开式逼近得到的。由于计算机本身并没有抽象的函数概念，它只能通过这种多项式的方法来表示和运算。另一方面，多项式展开的方式可以人为地控制，其精度和偏差也能做出相应的限制，因此，在实际情况中，这是非常快捷而有效的处理方法。

数字喷泉码的实现方案也不例外，其中也涉及到了一些不错的近似处理手段。然而在实践中，我们发现了 LT 编码中一个关于随机数产生的问题——重复关联，它是由近似算法造成的，并且直接导致了信息冗余、码字性能下降等危害性。

在本章中，我们将先指出重复关联问题的具体表现，然后对其进行理论分析，阐明其产生的根本原因，并推导出具体的严重程度。接着，我们设计出几种可行的改进方案，可以从根本上杜绝重复关联的发生。最后，经过仿真实验，验证了新方案的性能优越性。

3.1 LT编码中的重复关联问题

LT 码是数字喷泉码的核心，它采用一种随机性关联的方式来产生独立的编码，从而实现了无码率特性，具有真正意义上的码率控制能力。对于实际的应用，

专利[4]中提出了一套详细的 LT 编码实现方案，并给出了 RSD 度分布 $\mu(d)$的实现代码。

另一方面，针对 LT 编码中 OS 与对应 IS 的关联处理，文献[30]中提出了一种叫做随机性有限的 LT 编码（Limited Randomness LT Codes，LRLTC）算法，用于快速地产生关联信息，其算法流程如下：

i. 均匀随机地选取出两个独立的正整数 X、Y，并要求 $1 \le X \le K-1$，$0 \le Y \le K-1$；

ii. 对于每一个 OS，其第 i 个关联 IS 的序号为（$Y + iX \bmod K$），其中，$0 \le i < d$。

这里，K、d 分别表示 IS 集合的大小与当前 OS 的度数，另外，IS 的序号是从 0 开始一直到 $K-1$ 结束。

就随机关联操作而言，最容易想到的方法就是类似于一般的“扔色子游戏”：对于每一个 OS，先后进行 d 次“扔色子”操作，然后记下每次扔得的点数作为相应的关联信息。不过，这种方法未免过于复杂，因为每次处理的操作数都受到实际度数 d 的影响，d 越大时，其操作数也越大。

在 LRLTC 算法中，首先定义了两个随机的正整数 X、Y，然后通过它们的线性组合与求模运算，得到一个范围在 0 到 $K-1$ 之间的伪随机序列，从而可以近似作为关联信息，即所谓的“随机性有限”。这是一种线性同余的思想，它将无数个随机数的产生简化为 2 个基本随机因子的处理，从而有效地降低了操作数。另一方面，由于伪随机序列的产生主要用到了简单的加法运算，因此，这也是一种非常快捷的关联算法。

不过，该算法在实际应用中也导致了一个严重的隐患——重复关联问题。如图 3-1 所示，这里，虚线表示正常的关联情况，实线表示重复关联的情况。其中，$OS(0)$、$OS(1)$、$OS(3)$、$OS(4)$、$OS(6)$均为正常的 OS，它们都拥有规范的关联 IS；而对于深色标识的 $OS(2)$、$OS(5)$、$OS(7)$，它们都拥有重复的关联 IS，是不规范的编码符号。

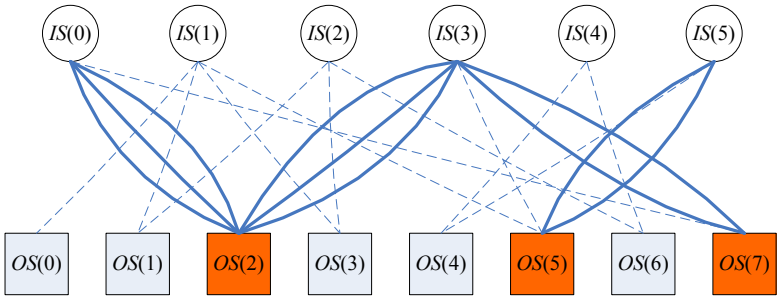

图 3-1　LT 编码中的重复关联示例

具体对应到前面的 LRLTC 算法来说，当 $X=3$、$Y=3$、$d=6$ 时形成了 $OS(2)$，它重复三次地与 $IS(0)$、$IS(3)$进行关联，从而导致了重复的信息，因此，其实际的值被简化为 $OS(2)=IS(0)\oplus IS(3)$；同理，当 $X=4$、$Y=5$、$d=4$ 时产生了 $OS(5)$，重复关联了两次 $IS(5)$，其实际值为 $OS(5)=IS(1)\oplus IS(3)$；当 $X=3$、$Y=3$、$d=3$ 时出现了 $OS(7)$，重复关联了两次 $IS(3)$，其实际值为 $OS(7)=IS(0)$。

显然，重复关联现象的出现违背了 LT 编码设计的初衷：由于同一个 IS 多次被关联，导致最终得到的编码结果包含了冗余信息，从传输上讲这是一种浪费，而对于译码器来说，无疑也是增加了算法复杂度的和接收开销；另一方面，从具体操作的角度来看，每两个相同的数（IS）经异或运算之后，其结果必为 0，因此，重复关联使得相应的 IS 信息被屏蔽（抵消了），导致了无用的操作。更重要的是，重复关联问题直接引起了度分布函数的改变，在图 3-1 中，由于重复关联的 IS 被抵消掉，使得 $OS(2)$、$OS(5)$、$OS(7)$的实际度数分别为 $d=2$、$d=2$、$d=1$，并非前面算法中给出的 $d=6$、$d=4$、$d=3$，原先本该属于后三个度数的概率区域也转移到了前两个度数上，使得低度数的概率增大，因此，这从本质上说，度分布函数被隐式地改变了。

从上面的分析可以看出，具有重复关联的 OS 是一种无效的编码符号，它不仅引入了信息冗余，也增加了不必要的运算开销，如果是大型的服务器，这种浪费也许还能维持，但对于一些功率受限的设备（比如手机终端、PDA 等）来说就尤为致命了。另一方面，从前面第二章中我们知道，度分布的设计是 LT 编码的关键，由于度分布的改变，其译码性能也必将受到影响，丧失了最初的有效性原则。因此，我们必须寻求一种方法来解决它。

3.2 重复关联的理论分析

经过前面的说明，我们知道了重复关联问题的危险性。下面，我们来具体分析一下无效符号产生的原因及其严重性。

3.2.1 产生原因

为什么会出现重复关联的问题呢？

首先，定性地说，这是因为关联操作中没有考虑到后选定的 IS 与先选定的 IS 可能出现雷同。换句话说，倘若每选定一个 IS 后，将其取出，用剩下的 IS 继续进行随机抽取，那么就不会出现重复关联的情况了。对于具体的 LRLTC 算法

来说，由于随机数 X 在规定范围内没有加以限制，其各种可能的情况都是有机会出现的，其中也包括了重复关联出现的情况，因此重复关联问题是在这种条件下无法避免的，它会以一定的概率发生。

之所以会出现这种情况，从数学的角度来说，是因为 X 与 K 拥有一些不等于 1 的公共因子；同时，还必须满足度数 d 不能太小的条件。

由于 X 和 K 之间存在着一些公共因子，所有从前面的线性求余算法可以推知，经过了周期长度为 $K / GCD(X, K)$的一系列操作之后（这里，$GCD(\bullet)$表示最大公约数算子），Y 将恢复到最初的值上来，从而产生了重复关联的现象。这里，由于公共因子不为 1，则 $GCD(X, K) \neq 1$，所以这个周期长度肯定小于 K，使得 d 个 IS 被分配完之前有可能出现重复；否则，如果 X、K 互素，即 $GCD(X, K) = 1$，周期长度等于 K，由于 $d \leq K$，这时就一定不会出现重复关联了。

另一方面，正如前面提到的，在经历完这个重复周期之前，如果 OS 的度数已经分配完毕，那么重复关联也不会发生。因此，还需要对 d 有所要求，必须让它足够大。具体来说，d 不能够小于 3（$d \geq 3$）。这里，我们考虑最极端的情况，即当 X 是 K 的因子，并且 $X = K / 2$ 时，至少还需要 $d = 3$ 次操作才能使 Y 恢复到最初的值上来，对应到图 3-1 中，这类似于 $OS(7)$的情况。

综上可以看出，在 LRLTC 算法中造成重复关联的根本原因在于度数 d 的取值，以及 X 和 K 之间的公因子情况。前者决定了是否存在重复关联的可能性（当 $d \leq 2$ 时，根本不会发生），而后者确定了具体的产生结果。此外，需要注意的是，d 的取值虽然只有下限，但它越大越容易发生重复关联的情况，因为如果 d 小于前面提到的重复周期，也是不会发生重复关联的。在图 3-1 中，对于正常的 $OS(6)$来说，假设 $X = 2$、$Y = 2$（还可能有其他情况，例如 $X = 4$、$Y = 4$），如果 $d = 4$，那么最终产生的 $OS(6)$也将是个无效符号（因为 $IS(2)$重复了）。

3.2.2 严重程度

通过前面的论述，我们已经知道，由于 LRLTC 算法中未加限制的随机数 X，以及客观上较大的度数 d，导致了重复关联现象的发生。那么，现在大家需要关心的问题是，这种情况发生的可能性有多大呢？虽然我们前面已经提到了重复关联的危害性，但是倘若它发生的概率不大，那么从一定意义上讲，这种不良影响是可以忽略不计的。

那么到底重复关联发生的可能性大不大呢？从前面第二章的说明中，我们知道，RSD 度分布的特点是：在某些较大度数（必要条件）的位置上，也会出现一些较高的概率；同时，对于度数 $d \geq 3$ 的情况，其概率区间也占了整个 RSD 度

分布中的绝大部分。因此，从表面上看，这个可能性应该是非常大的。

下面，我们还是以 LRLTC 算法为基础来做一下具体的定量分析。为了方便说明，我们先给出两个起始定义：

定义 1　对于任意的正整数 k，$M(k)$表示除了 1 和 k 本身之外，能够整除 k 的所有正整数因子的集合，即 $M(k)=\{m_i \mid m_i \mid k, 1<m_i<k\}$；进一步地，我们用 $N(M(k))$表示 $M(k)$集合的元素个数。

定义 2　对于 RSD 度分布，p_s 表示度数大于一个正整数 s 的概率，即

$$p_s = \Pr\{d > s\} = \sum_{d>s} \mu(d). \tag{3-1}$$

在式（3-1）中，$\mu(d)$表示 RSD 度分布函数，即 $\mu(d)=(\rho(d)+\tau(d))/\beta$[11]。这里，第一项 $\rho(i)$表示理想的孤波分布（Ideal Soliton Distribution，ISD），其表达式为

$$\rho(d) = \begin{cases} 1/K, & d=1; \\ 1/d(d-1), & d=2,...,K. \end{cases} \tag{3-2}$$

第二项 $\tau(d)$是对 ISD 度分布的修正，使其在实际应用中具有较高的健壮性，其表达式为

$$\tau(d) = \begin{cases} R/dK, & d=1,...,K/R-1; \\ R\ln(R/\delta)/K, & d=K/R; \\ 0, & d=K/R+1,...,K. \end{cases} \tag{3-3}$$

其中，R 表示译码器收到度 1 的 OS 集合大小，δ 表示译码失败的概率。最后的 β 是概率的归一化系数，即

$$\beta = \sum_{d=1}^{K} \rho(d) + \tau(d). \tag{3-4}$$

言归正传，经过详细的分析和推导，我们得出了以下结论。

结论 1　对任意的 X，所有可能让 Y 恢复到其初始值的长度周期均被包含在集合 $M(K)$中，并且 $M(K)$也正好是由这些可能的长度周期组成。注意，这里的周期都是指的最小正周期（下同）。

证：首先，如果希望 Y 能恢复到其初始值，则必须经历一个完整 K（或 K 的整倍数）长度的距离，这样才能使求模操作（mod）刚好抵消掉这个长度距离累加所造成的影响。

对任意的 X，它都可以表示为 $X = GCD(X, K)\cdot D$，其中 D 与 K 互素，即 $GCD(D, K)=1$。

为了使长度周期包含 K 中 X 所没有覆盖到的那部分公因子，以保证 Y 可以经历完整 K 长度的距离而产生重复，可以推知，该长度周期必须

为 $K / GCD(X, K)$；显然，这里的 $K / GCD(X, K)$是 K 的一个因子，即 $K / GCD(X, K) \in M(K)$。

需要注意的是，集合 $M(K)$中并不包含 1 和 K，因为这两种情况可以忽略不计：当 $K / GCD(X, K) = K$ 时，即 X 与 K 互素，此时的长度周期为 K，但由于 $d \le K$，从前面的论述可知，在走完一个周期前，关联度数已被分配完毕，不会出现重复；此外，由于 $X \le K - 1$，因此 $K / GCD(X, K)$也不可能等于 1。可见，这两种情况可以排除。

另一方面，对任意的 $m_i \in M(K)$，令 $X = (K / m_i)$（这里，X 还可以有其它形式），可以推出，$K / GCD(X, K) = m_i$，由于 $1 < m_i < K$，因此，X 是一个合理的选取值，同时，m_i 也是一个可选的长度周期。

（证毕）

结论 2　对于长度周期 n，即能够使 Y 至少经过 n 次的累加求模后，恢复到其初始值的概率为

$$\Pr\{nX \mid K | Y\} = \frac{\varphi(n)}{K-1}. \qquad (3\text{-}5)$$

这里，$\varphi(\bullet)$是初等数论中的欧拉公式，即 $\varphi(n)$表示所有小于 n 且与 n 互素的正整数个数。

证：如果 $nX \mid K$，那么$(Y + nX) \bmod K = Y$，可以推知，条件“$nX \mid K$”等价于“Y 至少经过 n 次操作后可恢复到其初始值”。

根据结论 1，倘若长度周期是 n，那么令 $X = (K / n) \cdot D'$，其中 $0 < D' < n$（因为 $1 \le X \le K - 1$）；这里，还必须要求 X 是唯一与 n 相对应的数，以保证对于不同的 n，都有不会重叠的 X，所以需要满足 $GCD(D', n) = 1$，使 n 与 D'不会相互抵消掉。此外，对于不同的 D'，X 可以随之取到不同的值。实际上，整数 D'的取值非常有限，为了保证与 n 互素的前提条件，它只能有 $\varphi(n)$种取值选择。另一方面，X 是均匀分布在 1 到 $K - 1$ 区间上的，因此，Y 经过长度周期 n 之后能够恢复到其初始值的概率为 $\frac{\varphi(n)}{K-1}$。

（证毕）

结论 3　对任意的 K，能够使 Y 恢复到其初始值，并产生无效编码符号的总概率为

$$P_{invalid} = \frac{1}{K-1} \sum_{i=1}^{N(M(K))} \varphi(m_i) \cdot p_{m_i}. \qquad (3\text{-}6)$$

证：根据结论 1，我们先将所有 X 可能的取值进行子集划分，令 F_i 表示一个对应于 m_i 的 X 取值子集合，其中 $m_i \in M(K)$，要求 F_i 中的元素均以 m_i

作为其长度周期。

根据结论 2，集合 F_i 的元素个数为 $\varphi(m_i)$，并且 X 被包含在 F_i 中的概率为 $\varphi(m_i)/(K-1)$。

进一步地，对于固定的长度周期 m_i，那么根据前面的论述可知，必须要求 $d > m_i$ 才能保证 Y 经历一个完整 K 长度的距离恢复到其初始值，从而导致重复关联的发生。

因此，对不同的 m_i 求和，得到产生无效编码符号的总概率为

$$\sum_{i=1}^{N(M(K))} \frac{\varphi(m_i)}{K-1} \cdot p_{m_i} .$$

（证毕）

经过一番论证，我们终于从理论上推导出了 LT 编码中基于 RSD 度分布的无效符号率，即结论 3 所示。从式（3-6）中的各项因子来看，正如前面 3.2.1 小节的定性分析所说，产生 LRLTC 算法中出现重复关联问题的主要原因，在于 X 与 K 之间存在的公因子（具体来说，是公因子的数目，它直接决定了 m_i 的个数），并且实际发生的概率还与当前 OS 的度数有关。在后面的小节中，我们还会根据不同的情况，通过具体的 RSD 度分布，计算出各种情况下的无效符号率并用图表进行说明。

此外，对于确定的 K，根据式（3-6）可以直接计算出相应的无效符号概率，因此，LT 编码器产生无效符号的数目直接与 OS 数目 N 成正比。而对于实际的应用来说，数据量通常是很大的，N 的取值自然就不小。这时，如果直接使用前面的 LRLTC 算法进行操作，那势必产生大量重复关联的无效符号，造成整个编码传输系统性能的下降。因此，必须寻求一种改善的方案，从根本上杜绝（或者适当、有效地限制）无效符号的产生。

3.3 去除重复关联的方案

为了解决重复关联所造成的问题，目前主要有两个方向上的思路：

- 以原 LRLTC 算法为基础进行改进。从前面的论述中可以看出，LRLTC 算法的症结就在于没有对相应的参数做出限制，使其取值过于随意，造成无效符号的产生。因此，从实际的参数入手，设计一种方案对其加以限制，这从实现上来说也会便于调整。
- 设计一种新的关联算法。既然 LRLTC 算法直接导致了重复关联，那么寻求新的解决办法也未尝不可。简单地说，由于 LRLTC 算法在随机关

联时没有考虑到可能重复的情况，因此，只要设计一种算法，对关联操作进行一定的限制，保证能够独立、唯一地选取 IS 即可。

此外，由于 K 和 d 都是在关联操作之前就已确定的参数，它们应当作为常量来看待。同时，从第二章中 LT 编码的论述中还可以看出，这两个参数与度分布密切相关，都是算法中很关键的因素。因此，不能随便考虑对这两个值进行修改，否则会影响到码字性能。

总的来说，消除重复关联的办法必须基于一种折中交换的思想。由于 LRLTC 算法本身已具备非常低的常数级运算复杂度 $O(1)$，因此，新的设计方案可能需要适当削弱原始方案的某些优点，以换取必要的改进机制。这种思想在实际应用中也很常见，虽然本质上只是一种交换，但对于特定的应用，它们可能对某些指标要求很高，因此必须做出权衡，牺牲某些东西以改善指定部分的不足。

还有一点需要阐明的是，在 LRLTC 算法中有一种特殊的情况，就是如果 K 是素数的话，它除了自身和 1 之外没有其它的因子。所以，K 与 X 也必然互素，从而导致长度周期 $K / GCD(X, K) = K$，由前面的知识可知，此时也不会出现重复关联。在实际应用中，根据信源的大小，可以选择不同的 K，但受到客观条件的限制，我们无法保证每次的 IS 划分刚好是一个素数。因此，设计一种通用的排除重复关联的方案才更具有实用性。

基于以上一些考虑，我们设计了几种可行的关联算法，可以完全杜绝无效符号的产生。接下来，我们将一一对其展开讨论。

3.3.1 基于抽样的算法

这是一种比较简单的方案，其基本思路是：设计一个随机抽样的事件，每次从 IS 集合中选取一个并记录，然后将剩下的 IS 重新组合，供下一次选取。由于每次选取之后，都会把相应的 IS 拿走，这样就避免了重复。因此，在进行了 d 次抽样之后，可以得到 d 个不同的 IS。

该算法的具体流程见图 3-2 所示：

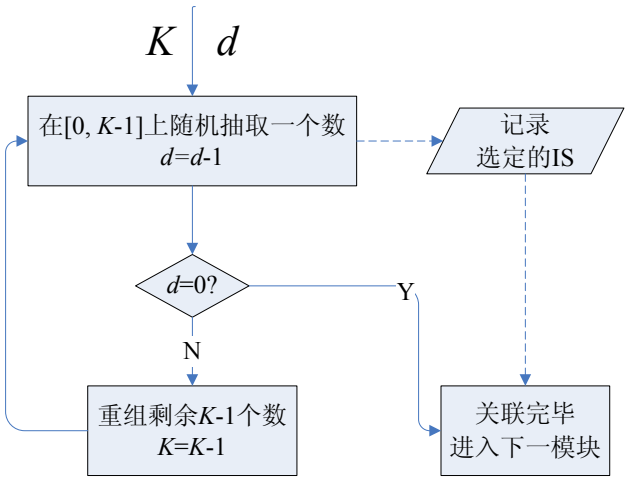

图 3-2 基于抽样的算法

i. 初始输入 K 和 d；

ii. 在区间$[0, K-1]$上随机抽取一个数并记录，同时更新 $d = d - 1$；

iii. 如果 $d = 0$，转到第 v 步；否则，转到第 iv 步；

iv. 重新组合剩余的 $K - 1$ 个数，并更新 $K = K - 1$，转到第 ii 步；

v. 关联完毕，结合已记录的 IS，进入到下一模块。

需要注意的是，在第 iv 步中用到了重组操作，因此，转到第 ii 步后，随机抽取的 IS 应记录其原始的序号，而非重组后的序号。还有一点，本来 d 和 K 的更新是可以同时进行的，但是如果已经到最后 1 轮抽样，更新 K 就没有意义了，这样的安排可以省掉一些开销。

在该算法中，对于给定的 d，先后需要进行 d 次随机抽样（包括 $d - 1$ 次重组），可见其运算复杂度为 $O(d)$。此外，在实际处理中，还需要额外的缓冲空间用以处理重组。

3.3.2 基于排序的算法

由于整个关联算法的主要任务就是从 IS 集合中先后选取出 d 个元素，既然涉及到先后问题，那么我们也可以考虑一种基于排序的方案，其基本思路是：先为每个 IS 分配一个随机数标签，然后利用这些标签进行排序（升序或降序均可），最后只需输出前 d 个 IS 即为所需的关联符号。

下面，我们以降序排列为例，说明该算法的具体流程见图 3-3 所示：

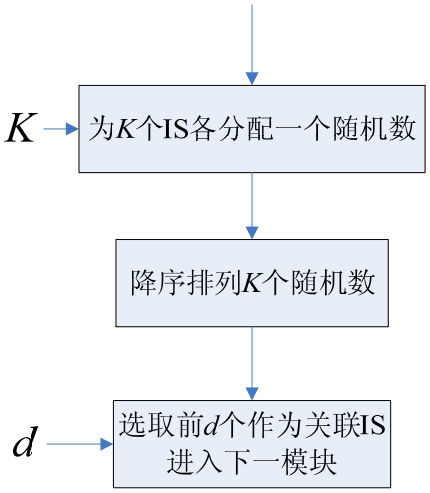

图 3-3 基于降序排列的算法

i. 随机处理，为 K 个 IS 各分配一个随机数；

ii. 对 K 个随机数进行降序排列，得到一个新的序列；

iii. 直接选取新序列中的前 d 项作为当前 OS 对应的关联 IS，并进入到下一模块。

在该算法中，每个 IS 对应一个随机数标签，而通过排序处理可以打乱原 IS 集合中的元素顺序，使其随机化；另一方面，即使随机数一样，最后排序的结果也不会出现 IS 位置重叠的情况，所以，这种算法可以在保证随机性的前提下也去除了重复关联。

复杂度方面，随机数分配需要进行 K 次，而排序操作，我们以快速排序为例，其平均复杂度为 $O(K\log K)$。从而可以推出，该算法的复杂度为 $O(K\log K)$，它与 d 无关，但是因为 $d \le K$，所以说，这个复杂度实际上是挺高的。不过，该算法中的随机处理可以选择任意范围内的随机数，只要能进行大小比较并完成排序即可，因此不用像前一种方案那么进行范围限制。还有一点需要说明，在处理排序时，由于 $d \le K$，因此并非每次都要对 K 个随机数进行全排列，可以只考虑前 d 个数的简化排序算法，从而进一步降低复杂度。此外，同前一种方案一样，该算法也需要额外的缓冲空间用以处理排序。

3.3.3 基于LRLTC的改进算法

由于 LRLTC 算法本身具有非常优越的性能，因此，以该算法为基础进行改进，这也是非常合理的方案。既然原算法的问题就出在几个参数的选取上，那么

我们可以直接考虑对参数本身来进行修正、加以限制，使其可以满足需要即可。从 3.2 节的理论分析可知，与重复关联密切相关的参数只有 K、d 和 X，正如前面所说的，K、d 两个参数应作为先验常量来处理（不能随便调整），因此，这个问题就化简为对参数 X 的修改上了。

基于以上分析，我们设计出一套具体的算法，见图 3-4 所示：

i. 均匀随机地选取出两个独立的正整数 X、Y，并要求 $1 \le X \le K-1$，$0 \le Y \le K-1$；

ii. 对于给定的 X 和 d，

如果 $1 < X < K-1$ 并且 $d \ge 3$，则定义长度周期

$period = K / GCD(X, K)$，

如果 $period \ge d$，转到第 iii 步；

否则，令 $X = X + 1$，回到第 ii 步；

否则，转到第 iii 步；

iii. 对于每一个 OS，其第 i 个关联 IS 的序号为（$Y + iX \bmod K$），其中，$0 \le i < d$。

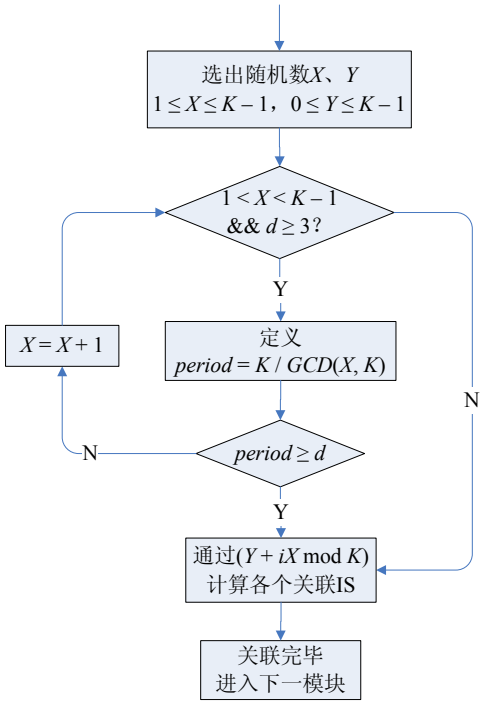

图 3-4　基于 LRLTC 的改进算法

在该算法中，第 i、iii 两步继续沿用了 LRLTC 算法，唯一不同的就是在第 ii 步中添加了 X 的修正机制。

在第 ii 步中，当 $X = 1$ 或 $X = K-1$ 时，由于 X 与 K 互素，其长度周期直接等于 K，由于 $d \le K$，那么这两种情况下根本不会出现重复关联的。所以，该算法修正的对象主要针对 $1 < X < K-1$ 这个范围来进行。同时，由前面的论述可知，

$d \geq 3$ 这个条件也是出现重复关联的必要条件。进一步地，对于度数 d 而言，如果它在一个长度周期 *period* 之内就完成了所有关联 IS 的分配，即 $period \geq d$，那么 Y 也不会恢复到其初始值并产生无效符号，于是，可以直接跳到第 iii 步去完成关联操作；否则，修改 $X = X + 1$，然后返回到第 ii 步重新进行判断处理，直到满足条件转到第 iii 步为止。此外，这里有个问题需要注意：如果 X 一直都不满足条件，那么是否会造成死循环而无法退出呢？答案当然是否定的，从上面的步骤可以推出，如果 X 不断地进行修改，那最终情况就是加到 $X = K - 1$，此时自然能够满足前面的判断条件，进而成功完成关联。可见，这种算法也能完全解决重复关联问题。

还有一点需要指出，该算法中对 X 的具体修改方式并不唯一，之所以选择 $X = X + 1$，是因为这种修改相对简单而有效。也可以考虑其他有一些办法，例如，可以使用对偶的公式 $X = X - 1$，容易验证这种方法也是可行的，如果 X 不断修正下去，那最终也会出现 $X = 1$ 的情况，此时是可以正常退出的。还可以考虑直接重新获取 X，当发现它不满足条件时，另行分配一个随机数来取代它。不过，这种方法不能有效地保证退出，如果每次重新分配的 X 都不满足条件（这种情况虽然少，但是却存在），那就可能长期死锁在这个上面，而前面两个公式都可以保证在 K 次修改以内完成关联。因此，这种方法的退出条件是基于一定概率的，而不像前面两种修改方案那么完整。

通过以上分析，我们可以发现，与原 LRLTC 算法相比，该算法多出来的开销主要在于最大公约数 $GCD(\bullet)$和修改公式的计算上。前者是在可能出现重复关联的假设下进行的，而后者则是在参数不满足条件的情况下进行的。所以，两者都是基于一定条件才会发生的事件，严格来说，其概率并不高，而且后者的可能性更小，因此它们所引起的复杂度改变也并不大。换句话说，该算法是在复杂随机化与快速随机化两种方式上进行了折中，既排除了无效符号，又保留了简化的思想，同时还尽可能小的限制了复杂度的增加，这在实际应用中具有十分独特的优越性。

3.3.4 算法比较

在前面的小节中，我们分别讨论了三种可行的改进方案。下面针对各项指标，我们对其一一进行比较：

- 随机性方面，显然，前面两种方案较高，因为它们都是从一般的随机序列中进行随机数产生的，具有一定的随机性保障；而对于第三种算法，由于它通过简化的处理降低了复杂度，相应地，也牺牲了一定的随机性

（即“有限的”）；

- 时间复杂度方面，第一种方案具有线性的复杂度 $O(d)$；第二种方案的复杂度为 $O(K\log K)$，这个相对较高，不太适合于实际应用；最后一种方案，其复杂度虽然不及原 LRLTC 算法的常数级那么好，但也还不至于到线性复杂度那么差，因此，它在这方面的性能优于前两者；
- 空间使用方面，前面两种方案都需要额外的缓冲进行处理，从而也带动了一系列的空间开销，而且第二种方案略大；与前两者相比，第三种算法相对消耗较小；
- 处理逻辑方面，前两种方案都比较简单，不过第二种方案的排序处理还需要更细致的算法处理；而最后一种方案则相对复杂，它需要更多的逻辑判断和流程控制开销。

在下一节里，我们分别针对这几种方案做了进一步的实现，并通过仿真数据进行了性能比较。

3.4 仿真实验

经过前面的讨论，我们先后分析了重复关联发生的原因，以及无效符号产生的概率。进一步地，通过分析和推理，我们提出了三种不同类型的解决方案，并分别做了比较。下面我们来进行一些仿真实验，以验证前面论述的观点及其正确性。

3.4.1 无效符号率

针对 3.2.2 小节中论述的无效符号概率，我们进行了两组计算实验。注意，这里的仿真均是统计意义上结果，即每个数据都是进行多次处理后取平均得到的，而并非单次测试的特定结果（下同）。

如图 3-5 所示，针对不同的 K（100 ~ 1000），我们分别通过式（3-6）进行了计算，并得出 RSD 度分布下相应的无效符号概率。需要说明的是，对于 RSD 度分布的具体实现方法，我们参考了专利文献[4]中的代码（下同）。从图中可以看出，无效符号产生的理论概率大致在 1% ~ 5%之间。此外，我们还利用原 LRLTC 算法进行了实践验证：给定 $N=K$，然后统计在不同 K 的条件下，各自产生的无效符号数所占总 OS 数目的比率。对比两条曲线，可以发现它们基本相似，并具有一致的趋势，这反映出了理论与实践的统一性原则。此外，还需要说明的是，

图中的曲线存在一些陡峭不平的情况，这是由于不同的 K 所包含的因子数目，以及其对应的度分布函数不同所造成的。

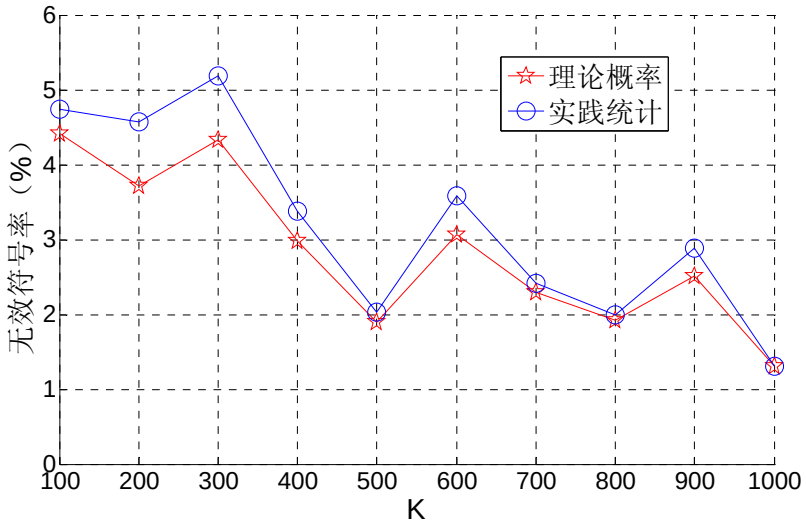

图 3-5　基于 RSD 度分布的无效符号率

如图 3-6 所示，针对不同的 K（100～1000），以及三种不同的 N（1.0K、1.5K、2.0K），我们分别统计了各种情况下的平均无效符号数目。从图中曲线的趋势来看，它们都与图 3-5 中的曲线相一致。此外，随着 N 的增大，对应的无效符号数目也不断增加，这意味着编码性能的进一步恶化。

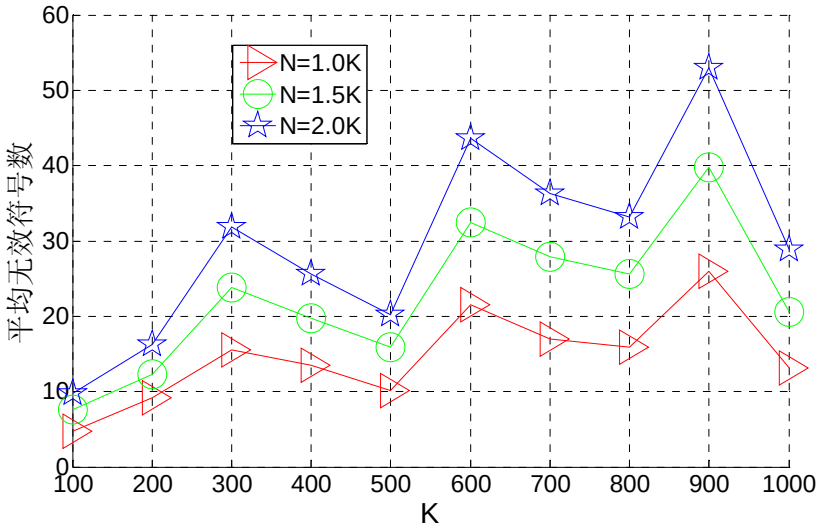

图 3-6　不同 N 下的平均无效符号数目

对于一般的互联网场景来说，其网络传输的丢包率通常都要求不高于 1%。而相比之下，我们计算出的无效符号率却都在 1%以上，这反映出该问题具有不可忽视的严重性。另一方面，从图 3-5 中的趋势来看，随着 K 的增大，无效符号率可能会有所下降；但是相应地，N 也会随之而增大，使得无效符号产生的绝对数目还是无法减小。实际应用中，从数据类型来看，N 的取值往往都很大，在这

种情况下，如果使用前面的 LRLTC 算法，那势必造成非常大的传输隐患。

3.4.2　各算法的改善对比图

针对 3.3 节提出的各种改善方案，我们以译码性能为例，进行了一组对比仿真实验。这里，我们首先定义一个性能参数如下

$$R = \frac{U}{K} - 1. \qquad (3\text{-}7)$$

其中，U 表示译码成功所用到的 OS 数目。因此，R 就表示在保证译码质量的情况下所需要的 OS 过载率，过载率的高低可以直接反映出编译码性能的优劣。同时，为了合理计算出过载率，在实现中，N 应取作任意大的数，以保证绝对能译码成功。

针对前面的四种方案（原 LRLTC 算法、基于抽样的算法、基于排序的算法、基于 LRLTC 的改进算法），我们分别进行了四轮仿真实验，并根据式（3-7）的定义，绘制出了相应的曲线。如图 3-7 所示，横轴仍然是 K（100~1000）这个条件。原 LRLTC 算法由于重复关联问题产生了无效符号，因此直接导致了其较高的过载率，也反映出其性能上的不足；而相对地，三种改善方案的过载率较低，并且其相互间的差异也不大。从图中曲线的趋势来看，随着 K 的不断增大，R 也在逐步缩小，但是无效符号的数目还是在进一步地增加，只是速率稍缓了一点。因此，这个问题还是不容乐观。

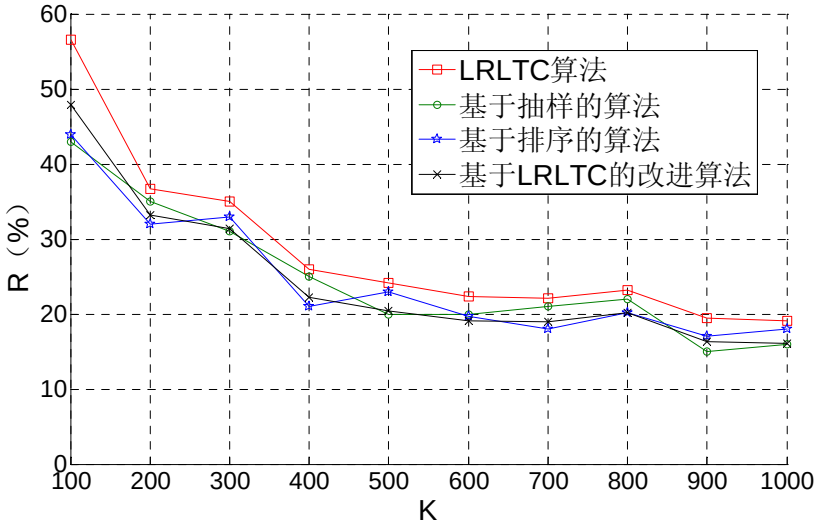

图 3-7　各方案的过载率比较

进一步的统计数据是：三种改善方案在 OS 上的平均节约数目在 50 个左右，其平均节约率也在 3.5%左右。需要指出的是，这不仅仅是接收开销上的节约，

它同时也减小了译码中的迭代运算量，提高了译码性能。另一方面，正如前面3.3.4 小节算法比较中所提到的，前两种方案在多次仿真实验中的时间开销较大，引起了一定的不便，因此，综合各项指标来看，我们发现基于原 LRLTC 算法的改进方案具有更好的平衡性与实用性。

3.5 本章小结

在本章中，我们围绕 LT 码在工程实现中的重复关联问题进行了研究与讨论。

由于原 LRLTC 方案为了提高运算效率，简化了关联方案，从而导致了重复关联的产生。这一方面增加了信息的冗余性，造成了不必要的开销；另一方面也改变了实际的度分布情况，引起了性能的下降。因此，必须解决这个棘手的问题。

通过分析，我们找到了产生该问题的原因，即度数 d 的取值，以及 X 和 K 之间的公因子分布情况。进一步地，我们从理论上推导出了实际 LT 编码中，基于 RSD 度分布下产生无效符号的概率，并阐明了影响它的主要因素。接着，我们从产生的原因入手，分别设计出 3 种可行的解决方案，并进行了相应的性能分析和比较。

最后，我们设计并实现了整个 LT 编译码系统。通过仿真实践，确认了实际应用中产生重复关联问题的严重性，并从对比的实验结果中，进一步验证了改善方案的优越性。

（注：本章的相关内容已发表到会议 WiCOM ’09 中并已录用。）

第四章 基于数字喷泉码的分布式编码方案

分布式信源编码（Distributed Source Coding，DSC）理论由来已久，这最早可追溯到 Slepian-Wolf 的无损分布式编码原理和 Wyner-Ziv 提出的有损分布式编码[31]。由于当时缺乏实际的应用场景，这些概念停滞了很长一段时间。近年来，随着科学技术的不断发展，以及人们对科技需求的日益提高，如何使网络中的节点（尤其是功率受限的设备）具有一定的计算能力，并能在较低复杂度的运算下进行编码传输，成为了一个比较热门的话题。DSC 的优势就在于其编码复杂度简单，这正好迎合了当前的需求。因此，DSC 的研究又获得了重生。

DSC 理论只是一种设计思想，而并非具体的技术方案，它通常需要一种实际的信道码作依托来进行构造。目前，已经实现并且可行的方案包括分布式 Turbo 码[32]、分布式 LDPC 码[33]等。数字喷泉码作为一种新型的信道码，自然也可以用到分布式场景中，文献[27]中就提到了一种基于 BSC 相关性的分布式 Raptor 码，并通过仿真验证了其可行性。

在本章中，我们以基本的 DSC 理论为基础，介绍了整个 DSC 体系，并重点围绕分布式 Raptor 码进行了研究。经过讨论，我们发现了度 1 符号（Degree-One Symbol，DOS）的重要性，以及 DSC 场景的特殊性，并设计出一种基于 DOS 的编码优化算法。最后，通过实验验证了该方案的合理性。

4.1 DSC理论

经典信息论[34]告诉我们，如果源数据自身存在某种程度的冗余性，那么可以通过一定的方法，在不丧失信息的前提下去除这些冗余，从而达到压缩的效果，这也是信源编码的基本思想。进一步地，把这种冗余的概念推广到多个信源中，则可以描述为各信源间存在着相关性，简单地说，即在已知某个信源信息的前提下，可在一定的概率下（不为零）推测出与其相关的信源信息。因此，利用这种冗余性，也可以进行数据的压缩。在实际处理中，这一般是用联合编码的方式来完成，通过解析信源间的相关性实现压缩。

DSC 理论指出：在发送端，无论各信源是否进行联合编码（可以是独立编码），经过传输后，在接收端都能通过联合译码达到一样的性能效果（即信息量相等），如图 4-1 所示。换句话说，联合编码并不是解析信源相关性的必要处理，并且独立编码也不会造成初始信息的流失。由于独立编码的运算复杂度要比联合编码低很多，因此在实际应用中，我们可以对信源进行相对简单的独立编码和传输，这样有助于能量的节约，非常适合于功率受限的网络终端。不过相应地，译码端的负担也会随之而增加，通常需要设计更为复杂的机制来解析信源相关性，并完成联合译码。总的来说，DSC 的方法并不是从根本上完全简化了整个编码体系，而是通过一种手段，将编码端的编码复杂度转移到了译码端，从而降低了编码开销而增加了译码的难度。这是一种折中的处理，但它正好适合于一些特定的场景，例如无线传感器网络。

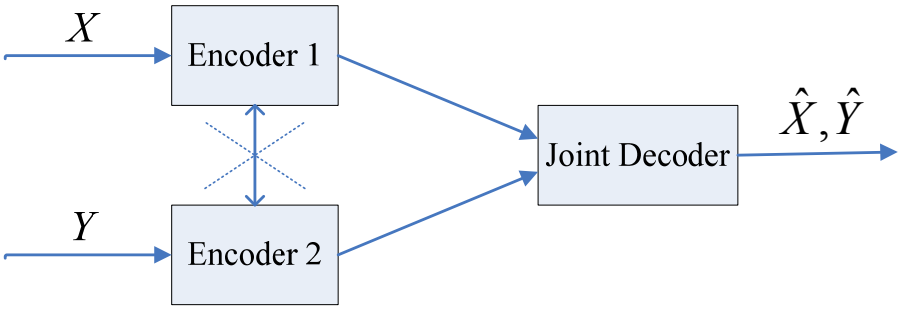

图 4-1 两信源的 DSC 系统[31]

图 4-1 中给出了两信源的 DSC 系统示例，在实际处理时，信源 Y 通常是进行自身的熵编码，其传输的信息量为 $H(Y)$；而对于信源 X，为了保证压缩的效果，其与 Y 传输的信息量之和不能超过它们的联合信息熵 $H(X, Y)$，所以，它必须按照 $H(X|Y)$进行独立压缩。这样，在接收端，译码器首先可以完全通过 $H(Y)$恢复出原始的 Y；然后，以 Y 作参考（边信息），利用其与 X 之间的相关性（包含在 $H(Y)$中），以及 X 发送过来的信息 $H(X|Y)$，共同完成对 X 的联合译码。从这里可以看出，由于 Y 自身的处理在整个过程中几乎是完全独立的，因此，DSC 系统的关键问题在于对 X 的压缩编码和联合译码，它们需要依托 X 与 Y 的相关性来完成。

那么，如何对 X 按 $H(X|Y)$进行压缩，并保证最后的译码质量呢？

在文献[35]中提出了一种基于 Syndrome 的 DISCUS（Distributed Source Coding Using Syndromes）方案，它先基于信源相关性，对 X 的符号空间进行划分，构建出不同的陪集并完成符号归类。然后，针对每个 X 符号，每次只需发送其对应的陪集序号即可。在译码端，根据收到的序号索引，并结合已复原的 Y，在相应陪集中寻找与 Y 最相似的 X 即为译码结果。

另外，还有一种基于 Parity 的实现方案，它先利用一种（系统的）线性信道码对 X 进行编码（这里，输入长度为 K，输出长度为 N），然后去掉其中的系统位部分，只保留 $N-K$ 长的 Parity 部分进行传输，由于 $N-K$ 往往小于 K，所以这也相当于一种压缩处理。在译码端，直接将恢复出来的 Y 作为 X 的系统位部分，并与接收到的 Parity 部分一起送入线性信道码的译码器进行对 X 的估计。该方案实际上是利用一种信道模型，对两个信源之间的相关性进行了模拟，即把 Y 看作是 X 经过一个虚拟的信道传输之后的结果。该虚拟信道的参数可用 X 与 Y 之间的相关性来描述，这样就可以直接通过原先的信道译码来恢复 X 了。前面提到的基于 LDPC 码的分布式方案[33]，就是基于这种设计思路来实现的。

上面这两种思路是目前比较主流的 DSC 设计方法，其本质上是一样的，不过也各具特色。在 DISCUS 中，主要的问题在于陪集的划分，因此其前期的编码预处理相对复杂一些，但是相应的译码比较容易，只是简单的匹配处理。在基于 Parity 的算法中，其编码端是不用任何特殊处理的，只需发送 Parity 部分即可；而在译码端，由于引入了虚拟信道，它对该信道的参数（相关性建模）依赖性较高，因此其主要运算量集中在译码器上。

此外，由于分布式思想的独到之处，该方案可以适用于许多具体的应用场景。将 DSC 应用于视频处理中，便出现了分布式视频编码（Distributed Video Coding，DVC）[36]。对于视频序列来说，由于前后连续的图像帧之间存在着很大的时间相关性，因此可以对其进行建模，把各帧图像模拟成虚拟的相关信源，这样正好符合 DSC 设计的前提条件，能够进行相应的压缩处理。在文献[37]中，提出了一套具体的 DVC 方案，可以解析视频帧之间的相关性，并具有很高的实用价值。

总的来说，分布式场景的前景非常可观，随着信息网络的逐步扩大，这种设计思想必将大受青睐。

4.2 DSC场景下的Raptor码

在文献[27]中，作者以 Parity 实现方案为基础，提出了一套基于 BSC 相关性的分布式 Raptor 码，其系统架构见图 4-2 所示：（注意，这里我们都是用两信源的场景来进行讨论，并且均以系统 Raptor 码作为实际的应用方案，其中的线性预编码采用 LDPC 码，下同）

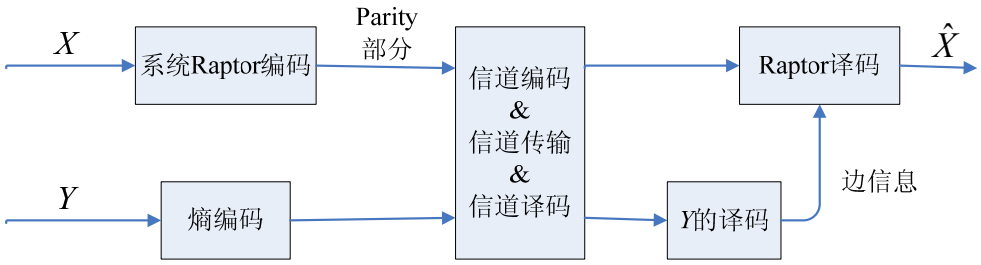

图 4-2　分布式 Raptor 码系统框图

i.　X、Y 分别进行 Raptor 编码和熵编码，其中，X 只输出 Parity 部分；

ii.　中间先后经过信道编码、传输、信道译码；

iii.　通过 Y 的译码器，先得出边信息 $\hat{Y}$，并传入 X 的 Raptor 译码器；

iv.　结合 Parity 部分与边信息，最终得出结果 $\hat{X}$。

此外，在文献[27]中，不同于一般的“分离式”Raptor 译码方案[18]（即先进行 LT 译码，然后在进行 LDPC 译码），作者基于置信传播的思想，提出了一种联合的迭代译码算法，将级联的两层编码统一起来，并通过仿真验证了其优越性。如图 4-3 所示，这里分别列出了 OS、IS、CS（Check Symbol，即 LDPC 码中的校验符号）三种节点及其之间的关联图。

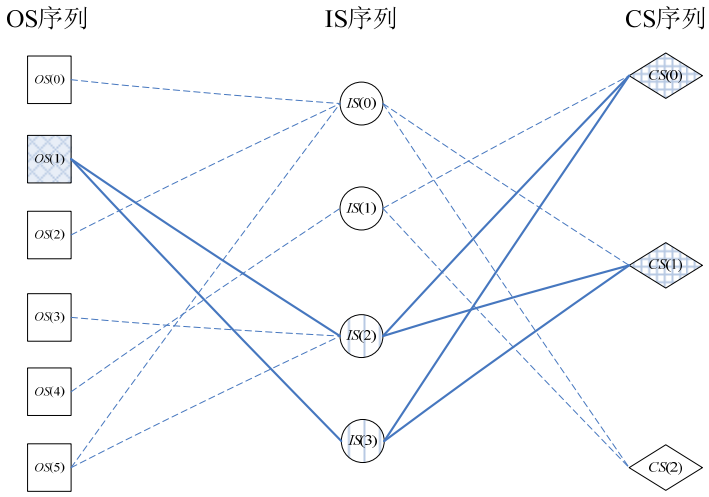

图 4-3　联合译码中的信息流向示例

我们以 $OS(1)$为例，简单说明一下每一轮信息传递的过程（图中的实线部分）：

i.　首先，$OS(1)$从关联的 $IS(2)$、$IS(3)$中收集信息，并传递给它们；

ii.　然后，$IS(2)$、$IS(3)$收集关联 OS 传来的信息，这里还包括 $OS(3)$、$OS(5)$，并将其传递给关联的 $CS(0)$、$CS(1)$；

iii.　接着，$CS(0)$、$CS(1)$收集其关联 IS 的信息，并回传给 $IS(2)$、$IS(3)$。

最后，如果迭代结束，则综合 $IS(2)$、$IS(3)$的信息来判定 $OS(1)$的估计值。

这里，需要注意的是，虽然图中只提到了几个直接的关联符号，但实际上，

整个过程牵扯到几乎所有的 OS、IS 和 CS，这样才能达到真正置信传播的目的。

下面，根据文献[27]中的论述，具体的迭代更新公式如下（注意，与前面的引用不同，该文献中的所以符号的序数是从 1 开始的，不是 0）：

LT 部分：

$$\tanh(\frac{\mu_{o\to i}^{(l)}}{2}) = \tanh(\frac{Z_o}{2}) \cdot \prod_{i' \neq i} \tanh(\frac{\mu_{i'\to o}^{(l)}}{2}), \quad o = 1, ..., K. \tag{4-1}$$

$$\tanh(\frac{\mu_{o\to i}^{(l)}}{2}) = (1-2p_o) \cdot \prod_{i' \neq i} \tanh(\frac{\mu_{i'\to o}^{(l)}}{2}), \quad o = K+1, ..., N. \tag{4-2}$$

$$\mu_{i\to o}^{(l+1)} = \delta_{ldpc}^{(l),i} + \sum_{o' \neq o} \mu_{o'\to i}^{(l)}, \quad i = 1, ..., n. \tag{4-3}$$

其中，$\mu_{o\to i}^{(l)}$、$\mu_{i\to o}^{(l)}$ 分别表示第 l 次迭代时，o 传递给 i（i 传递给 o）的信息（LLR）；Z_o 表示 o 的初始 LLR（参见式（2-3））；而 p_o 表示 Parity 中 o 的符号值；$\delta_{ldpc}^{(l),i}$ 表示第 l 次迭代时，从 LDPC 部分传来的 IS 信息；K、n、N 表示信源长度、IS 长度和 OS 长度。

另外，式（4-1）、式（4-2）是每次迭代的起始公式，其相应的起始条件为：

$$\mu_{i\to o}^{(0)} = 0, \quad i = 1, ..., n. \tag{4-4}$$

LDPC 部分：

$$\mu_{i\to c}^{(l)} = \begin{cases} \delta_{lt}^{(l),i}, & l = 0 \\ \delta_{lt}^{(l),i} + \sum_{c' = c} \mu_{c'\to i}^{(l-1)}, & l \neq 0 \end{cases}, \quad i = 1, \ldots, n. \tag{4-5}$$

$$\tanh(\frac{\mu_{c\to i}^{(l)}}{2}) = \prod_{i' \neq i} \tanh(\frac{\mu_{i'\to c}^{(l)}}{2}), \quad c = 1, \ldots, n-K. \tag{4-6}$$

其中，$\mu_{i\to c}^{(l)}$、$\mu_{c\to i}^{(l)}$ 分别表示第 l 次迭代时，i 传递给 c（c 传递给 i）的信息；$\delta_{lt}^{(l),i}$ 表示第 l 次迭代时，从 LT 部分传来的 IS 信息。

中间 IS 部分：

$$\delta_{lt}^{(l),i} = \sum_{o} \mu_{o\to i}^{(l)}, \quad i = 1, ..., n. \tag{4-7}$$

$$\delta_{ldpc}^{(l),i} = \sum_{c} \mu_{c\to i}^{(l)}, \quad i = 1, ..., n. \tag{4-8}$$

经过 L 次迭代后，最终的系统位符号 X_o 的 LLR 计算公式为：

$$LLR\_X_o = Z_o + 2\tanh^{-1}\left(\prod_{i} \tanh(\frac{\mu_{i\to o}^{(L)}}{2}) \right), \quad o = 1, ..., K. \tag{4-9}$$

（在后面的论述中，我们都将默认使用这种联合迭代的译码算法。另外，这里所有似然比均默认为偏 0 型 LLR。）

正如前面提到的，基于 Parity 的实现需要对实际的相关性进行建模，估计虚拟信道的实际参数。由于该文献中使用了 BSC 相关性，因此它体现在估计出的差错概率 p 上。另外，从式（4-1）可以看出，这里的信道相关性参数 p 主要影响到 Z_o，而与其他操作没有关系。因此，除了 BSC 相关性之外，我们还可以考虑其他一些情况，例如 AWGN（Additive White Gaussian Noise）相关性，只需重新计算 Z_o，并带入前面的迭代公式即可。在后面的小节中，我们还会针对这两种情况进行仿真对比。

4.3 基于DOS的优化设计

在前面的论述中，我们介绍了基本的 DSC 理论，并针对具体的实现方案进行了详细的说明。接下来，还是以分布式 Raptor 码为基础，我们将进一步围绕其中地位非常特殊的 DOS 进行讨论。

4.3.1 DOS的重要性

在第二章对 LT 码的论述中，我们就提到了在 MP 译码算法中扮演重要角色的 DOS，它是触发译码进行的先决条件，由于其只有一个关联 IS，因此可以直接计算出对方，并进一步把这种结果传递出去，从而形成一个“链式反应”。同样的，在 Raptor 译码的 BP 算法中，DOS 的作用仍然是非常关键的：

一方面，它也是触发译码的必要条件。从上一小节中可知，由于初始条件是所有 $\mu_{i\to o}^{(0)}=0$（式（4-4）），因此，如果当前 OS 的关联数目大于 1，那么根据式（4-1）、式（4-2）计算，后面的求积项必为 0，其结果则是所有的 $\mu_{o\to i}^{(l)}=0$，即传递了空信息，这样会使得整轮迭代没有意义。所以，必须要求有 DOS 的存在，只有它能传递出非零的信息，从而经过多次迭代，激发其他 OS 的关联能力，达到置信传播的效果。

另一方面，DOS 的正确与否，也直接关系到译码的结果。在删除信道中，由于译码端只有接收和丢弃两种选择，因此几乎不用考虑错误 OS 的情况。但在一般有噪信道下却不同，需要考虑符号差错的问题。如图 4-4 所示（这里省略了 CS），$OS(0)$、$OS(2)$、$OS(4)$都是 DOS，容易验证，如果它们都是正确的，那么译码结果也是正确的；另外，如果其中有一个错误，例如 $OS(2)$，那么在其它符号正确的情况下，也能保证译码结果正确；其它情况，译码正确的可能性都不大。可见，较多的正确 DOS 能够修正错误的符号，使最终译码正确的可能性提高。

需要注意，实际上译码结果还要受到其他符号正误性的影响，如图中的 $OS(1)$、$OS(3)$，这是它们共同作用的结果。

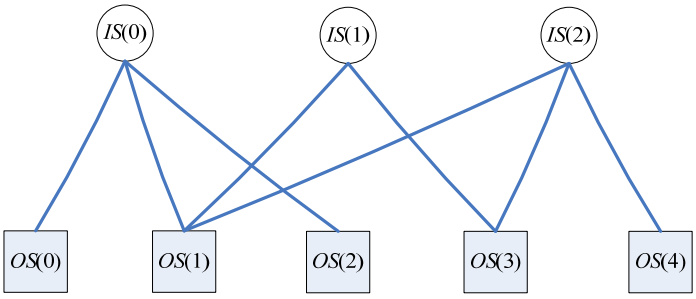

图 4-4　DOS 正确性影响的示例

总的来说，DOS 在译码中的作用非常重要，在后面我们还会进一步讨论关于它的特殊处理方案。

4.3.2　DSC场景下Parity的特殊性

在 DSC 方案中，实际发送的数据只有 Parity 部分，由于这只是信源编码（只考虑有效压缩），因此我们完全可以不考虑传输的可靠性问题（假设是理想信道，或者有完美的信道码来支持），将译码端接收到的 Parity 部分直接当作正确的数据来处理，而唯一需要关心的是利用边信息进行的虚拟信道译码。

既然 Parity 中的符号可认为是正确的，那么它们就应该具有绝对的似然比。在式（4-2）中，等号后面第一项$(1 - 2p_o)$，它在本质上其实是与式（4-1）中的$\tanh(Z_o / 2)$相一致的：由于其 LLR 为无穷大（正无穷或负无穷），所以由式（2-6）可知，它经过$\tanh(\bullet / 2)$运算之后的结果是± 1；另一方面，p_o 的值只可能是 0 或 1，因此$(1 - 2p_o)$的值也是± 1。由此可见，信源 X 中的符号在 Parity 或系统位中这两种情况的区别是：前者是无失真传输的，因此其相应的 LLR 为无穷大，具有绝对的置信能力；后者是经虚拟信道传输的，是一个有限的值，它需要靠 Y 来进行估计，而且该估计值还不能保证 100%的正确。这便是 DSC 场景下 Parity 的特殊性。

通过上一节的论述，我们知道了 DOS 的重要性，那么如果结合刚才提到的 Parity 特殊性，是否可以得出什么有用的结论呢？基于这样一种思路，我们认为如果在编码时，可以把 DOS 尽可能多的放到 Parity 中，那势必会对译码结果起到良好的作用。当然，这里的前提条件是不影响到码字原先的性能特点。

首先，我们来分析一下，如果一个 DOS 处在 Parity 中，那么它会对译码产

生什么影响呢？由于 DOS 是触发译码的关键，因此倘若能保证绝对正确的 DOS，那将是个很好的开端，并有助于迭代算法的良性发展。从这个意义上说，如果所有 DOS 都在 Parity 中，那整个置信传播过程也将非常顺利地进行下去。

另外，如果一个非 DOS 符号被分配到 Parity 中，那又是怎样的情况呢？这里，由于非 DOS 符号都有不止一个的关联 IS，它的实际取值会受到这些关联 IS 的共同作用，虽然它自身的 LLR 是无穷大，但却无法将这种绝对的判定能力传递出去。实际上，由于受到周围符号的干扰，这类 OS 的置信能力还可能被其它错误的符号所“腐蚀”，从而“稀释”掉其无穷大的 LLR。因此，这种分配方式具有一定的隐患。

在图 4-4 中，如果 *OS*(3)在 Parity 中，那么它传递给 *IS*(1)的信息会直接受到 *IS*(2)的影响，当 *IS*(2)的信息正确时，才能保证传给 *IS*(1)的信息正确；相反，*OS*(0)则不同，由于只有一个关联 IS，所以它的绝对置信信息可以准确的“传达”给 *IS*(0)。

从译码的角度来看，把 DOS 放到 Parity 中主要会对两个方面造成影响：一个是 BER（误比特率），试想如果 Parity 中的符号全是高度数（多个关联）的 OS，那么极端的情况下，由于受到系统位错误 OS 的干扰，这些高置信能力的 OS“全军覆没”，使得最终译码效果极差；另一个指标就是译码迭代的次数，由于 DOS 在 Parity 中，形成了良好的初始状况，因此，在一般情况下需要多次迭代来反复修正的操作可以适当地减少。

当然，如果 DOS 在系统位中，它也可能是在一定概率下是正确的。不过，正如式（4-1）所示，这种情况中其 LLR 是一个有限值，因而置信能力也是有限的，不及其在 Parity 中那么影响深远。

对于原方案来说，由于 Parity 所占的比例较小，DOS 在其中的数目也非常有限。因此，如果能够把 DOS 分配到 Parity 中，这将是一个不错的选择。在后面的小节中，我们会设计出一种编码优化方案，在不改变原码字性能的基础上，将 DOS 转移到 Parity 中，并通过仿真验证了我们的假设。

4.3.3 基于DOS的Parity优化方案

把 DOS 分配到 Parity 中，这从实现的角度讲，主要问题集中在 LT 编码矩阵的处理上。由于每个 OS 对应到编码矩阵中的一行，因此，这个问题就转化成了如何把重量为 1 的行转移到矩阵底部（底部对应 Parity）。

首先，一种简单的处理办法是：先按原方案产生编码矩阵 G_{LT}，然后对其中的每一行按度数进行降序的重排，这主要涉及到度数的比较和矩阵行交换操作。

不过，这种方法的效率很低，一方面排序所需要的复杂度为 $O(N\log N)$，比较高；另外，我们的目的只是要把 DOS 转移到底部，而没有必要“惊动”其他的行，或者尽可能少的去调整，这只会徒劳地增加开销。因此，该方法付出了非常大的代价，也造成了很多没有意义的操作。

基于这种考虑，我们重新设计了一种方案，见图 4-5 所示：

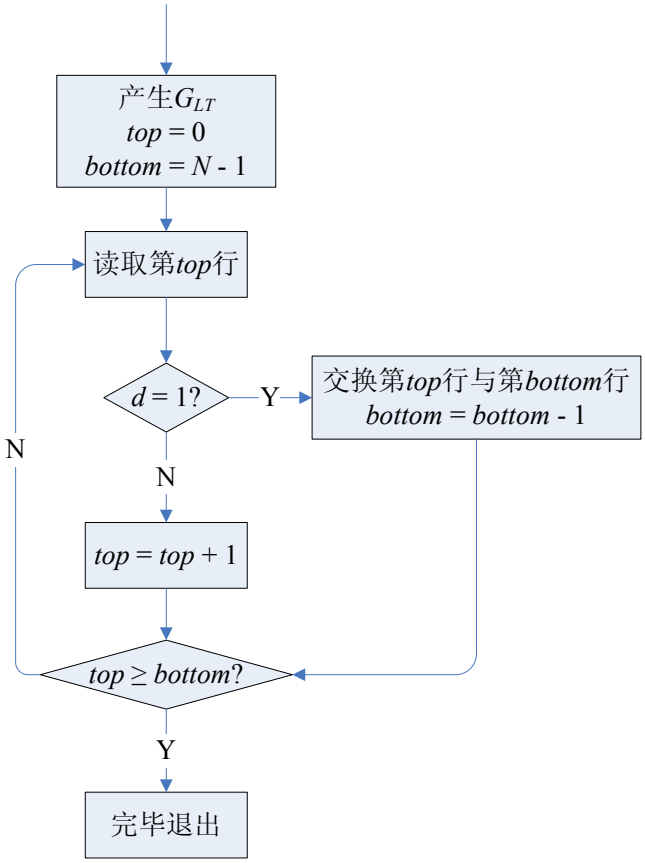

图 4-5　一种基于 DOS 的优化方案

i. 初始化，产生 G_{LT} 矩阵，并定义两个变量 *top*、*bottom*，对应 G_{LT} 中的第一行和最后一行；

ii. 读取第 *top* 行，判断其对应 OS 的度数：如果 $d = 1$，则转到第 iii 步；否则转到第 iv 步；

iii. 交换 *top* 与 *bottom* 两行，更新 *bottom* 到上一行，转到第 v 步；

iv. 更新 *top* 到下一行；

v. 判断是否遍历完毕：如果 $top \geq bottom$，转到第 vi 步；否则，回到第 ii 步；

vi. 处理完毕，退出。

该算法继承了排序的思想，对于给定的 G_{LT}，从上到下进行行遍历，每次通过交换将 DOS 相应的行转移到底部，从而达到我们的目的。需要注意的是，在

第 iii 步中，行交换之后只更新了 *bottom*（而并没有包括 *top*），这是因为原方案产生的 G_{LT} 在底部也可能存在 DOS 相应的行，即 *bottom* 可能也对应一个 DOS，故交换之后，还是要从原 *top* 的位置往下进行操作。此外，由于该算法并没有对整个矩阵进行行重排，因此，不难看出其复杂度是线性级别的 $O(cN)$，这里 c 表示 DOS 的出现概率。

上面的方案是基于既定 G_{LT} 来处理的，而我们需要对 G_{LT} 进行行序的改造，才能得到最终的目标矩阵。可见，最初的 G_{LT} 并不是我们的理想矩阵。那么，为什么我们不直接在产生 G_{LT} 时就对其进行约束，从而直接得到希望的矩阵呢？

基于这种思路，我们又设计了一种新的方案，见图 4-6 所示：

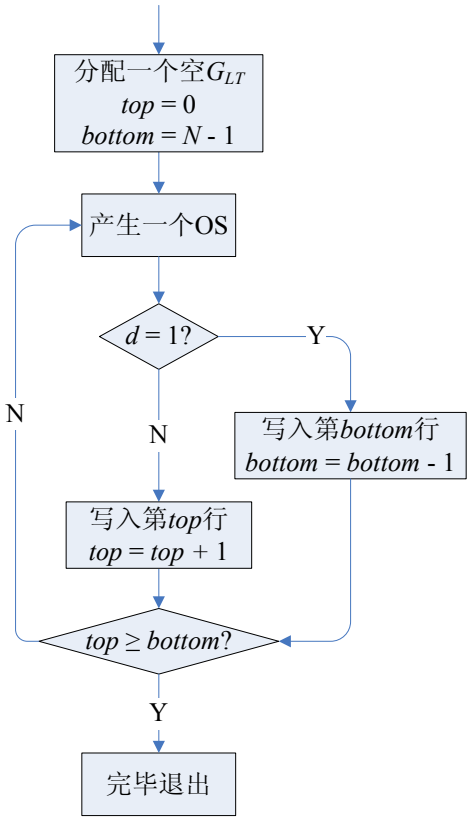

图 4-6　一种新的基于 DOS 的优化方案

i. 初始化时，只需定义两个变量 *top*、*bottom*，并分配一个空矩阵 G_{LT}；

ii. 按正常方式产生一个 OS，判断其度数：如果 $d = 1$，则转到第 iii 步；否则转到第 iv 步；

iii. 将当前 OS 信息写入到 G_{LT} 第 *bottom* 行（即对应 IS 上的 0 或 1），并更新 *bottom* 到上一行，转到第 v 步；

iv. 将当前 OS 信息写入到第 *top* 行，并更新 *top* 到下一行；

v. 判断是否遍历完毕：如果 $top \geq bottom$，转到第 vi 步；否则，回到第

ii 步；

vi. 处理完毕，退出。

在该算法中，我们直接在 G_{LT} 产生时就开始对 DOS 进行处理，每次遇到 DOS 则将其信息写入到矩阵底部，而其它 OS 则依次从顶部开始往下写入，这样，也可以达到最终的要求。从流程上看，该算法的复杂度也是线性的，与上一种算法差不多。但是，它把行调整的操作提前到 G_{LT} 的产生过程中，省掉了既定 G_{LT} 生成所带来的一部分写入开销。因此，这种方案在实现上的效率更高，具有更好的实用性。

这里，还需要考虑一个前提，即这种改进是否会影响原码字的性能？答案当然是否定的，正如前面关于 LT 码的论述中所提到的，编码器是独立产生每个 OS 的，其相互间并没有什么依赖关系，而每个 OS 正好对应到编码矩阵中的每一行，它们之间的交换、移动不会改变码字的性能，因此这种优化方案除了牺牲一点运算复杂度，并没有其他什么开销。

此外，可能有人会担心一点：当进行系统码转化时（参见 2.2.2 小节），会用到矩阵的行交换；那么，就算前面的处理把 DOS 移到了 Parity 中，由于系统化的行交换，这是否又会把它们调回到系统位中呢？如果这样的话，那整个优化就没有什么意义了。这种想法是有道理的，不过这种情况的可能性是比较小的。由于 Parity 在整个码字中的比例较小，同时 DOS 在度分布中的概率也非常低，因此 Parity 中的 DOS 数目对于整个 N 长的码字来说是很小的一部分，并且处于矩阵的最下方，很难会有“上调”的机会。当然，如果实在出现了这种情况，那大可重新生成一次 LT 编码矩阵来处理。客观来说，我们提出的方案是一种最大限度的尝试，它只能保证尽可能多的 DOS 分配到 Parity 中。

4.4 仿真实验

在本节中，我们将分别针对前面几部分的内容进行仿真，并通过实验结果来验证相应方案的合理性与可行性。

首先，根据 4.2 节中的介绍，我们分别针对 BSC 相关性与 AWGN 相关性，在 DSC 场景中进行了 Raptor 码实验。如图 4-8 所示，为了便于说明，我们统一采用了条件信息熵作为横轴，以反映 X 与 Y 之间的相关性。参数方面与文献[27]类似，即 $K = 10000$，$N = 15000$，$n = 10204$，统计块数为 1000，译码迭代次数为 100。同时，LT 部分的度分布采用了（下同）：

$$\Omega(x) = 0.008x + 0.494x^2 + 0.166x^3 + 0.073x^4 + 0.083x^5 + 0.056x^8 + 0.037x^9 + 0.056x^{19} + 0.025x^{65} + 0.003x^{66}. \quad (4\text{-}10)$$

而 LDPC 部分的采用了规则的(4, 200)LDPC 码。这里，LDPC 码的具体实现我们还参考了网站[38]中的生成方式（下同）。此外，AWGN 相关性的初始 LLR 公式为：（对应式（4-1））

$$Z_o = -2y/\sigma^2 \quad (4\text{-}11)$$

从图中可以发现，两种情况的曲线比较接近。不过，由于 BSC 相关性在本质上相对单纯一些，因此其性能会略高于 AWGN 相关性的情况。

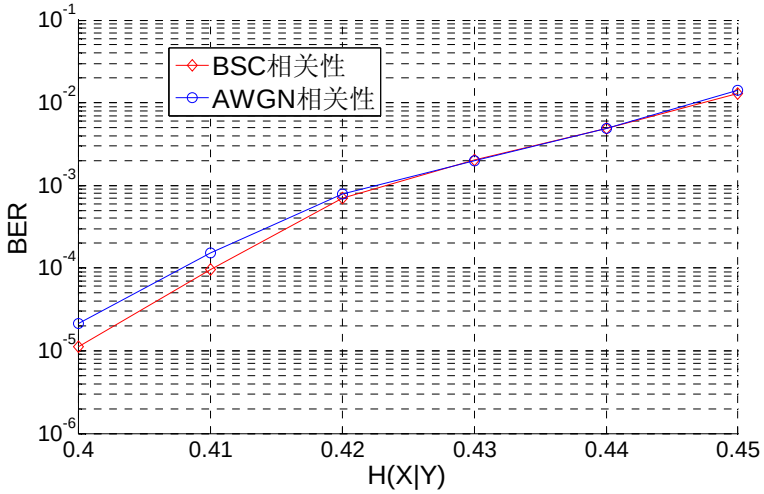

图 4-8　DSC 下的 Raptor 方案

进一步地，针对前面关于 DOS 的论述，我们基于 BSC 相关性，分别进行了三组实验：通过修改算法，把 DOS 按三种不同的方式分配到码字中；这里，对于 DOS 在顶部的情况，其产生方法与前面 DOS 在底部的算法类似，即通过判断，按升序进行度数排列，从而把 DOS 转移到顶部。此外，其它参数与上一实验相同。如图 4-9 所示，三种方案的 BER 曲线接近，不过改进后的方案性能更好一些。通过分析，我们发现，由于虚拟信道参数 p 比较小，使得整个码字中错误的符号也较少，因此造成了三条曲线之间的差异并不特别大。

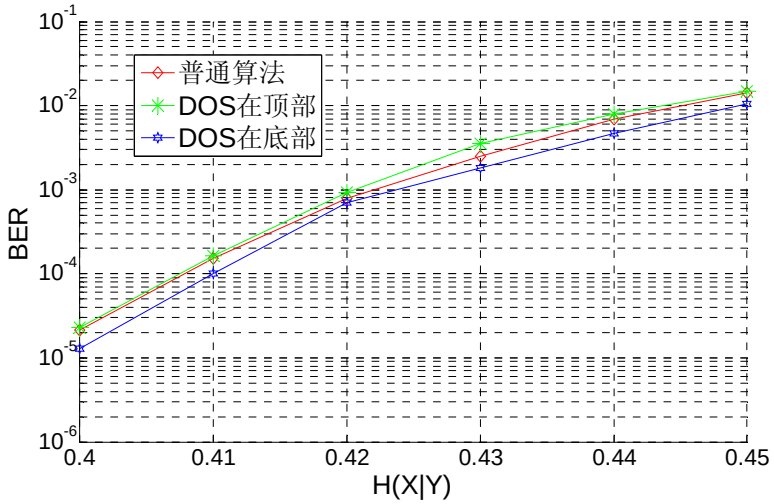

图 4-9　不同分布下的 DOS 对译码性能的影响

另一方面，我们还对迭代次数进行了对比统计，在译码成功的情况下，DOS 在顶部、普通算法、DOS 在底部这三种方案的平均最低迭代次数为 102.7、99.6、94.8。可见，改进后的方案在这个指标中也具备一定的优势。总的来说，新的方案虽然并不具备特别明显的优势（这也跟具体参数有关），但其至少不是一个坏的选择，它可以尽可能高效地保证译码质量。

4.5　本章小结

在本章中，我们围绕分布式场景下的数字喷泉码进行了讨论。

首先，由于 Raptor 码内含有一定抗噪能力的线性预编码（如 LDPC 码），使得它可以对一般的有噪信道产生纠错能力，因此它也具备了一般传统信道码的特点。另一方面，在 DSC 场景中，通常需要一种具体的信道码作为依托来实现。于是，基于以上两点的考虑，利用 Raptor 码来设计一套分布式方案也自然成为一种可能的尝试。因此，文献[27]提出了一套基于 DSC 的 Raptor 码，并验证了其可行性。

进一步地，我们诠释了 DOS 在数字喷泉码中的重要性，并结合 DSC 场景中 Parity 的特殊性（无穷大的 LLR）进行了分析，提出了将 DOS 转移到 Parity 中的优化性设想，并在不影响原码字性能的基础上，设计出一种具体的编码算法。通过实验结果可以看出，该算法具有一定的优越性。

第五章 结束语

数字喷泉码是一种前向纠错码，它可以产生无限的输出符号、灵活地进行码率控制（无码率特性），所以能够很好地解决传统删除码与互联网协议的问题。数字喷泉码无需反馈信道，它采用了一种单向的异步传输机制，具有高效、低延迟的性能特征。由于其编译码复杂度简单，因此也很容易在一些低功率的平台上实现。可以说，它既保证了传输的可靠性，同时又满足了应用中的有效性原则。

由于数字喷泉码的种种优势，使它具有非常广泛的应用场景，这包括四种基本的传输系统以及其衍生出的各种变体。随着一批批专家学者的不断努力，数字喷泉码已经由最初的理论研究领域进入到实用的领域。迄今为止，美国的 Digital Fountain 公司先后发布了一系列具体的相关技术专利；同时，它还备受国际社会的关注，一些标准化组织已将其纳入了相关协议之中，包括 3GPP、DVB、IETF 等。另一方面，数字喷泉码也受到其他研究者的青睐，包括分布式场景、认知网络等领域。可见，数字喷泉码已逐步成为一项成熟且实用的差错控制技术，其发展前景必将无限光明。

在本文中，我们围绕数字喷泉码，从理论基础到实现方案，分别进行了充分的说明，并总结了当前存在的问题与研究现状。进一步地，我们从 LT 编码技术入手，阐明了实现上的重复关联问题，并从理论上对其做了详细地分析和讨论，进而提出了有效的改进方法。另一方面，针对分布式的应用场景，我们以系统的 Raptor 码为例进行了仿真，而且通过分析，设计出一种基于 DOS 的性能优化方案。

5.1 完成的工作与成果

通过前面各章节的论述，我们已对数字喷泉码有了比较全面的认识，包括其优势与缺点，以及相关的应用场景等。下面，我们具体来归纳一下本文的工作成果：

- 简要介绍了数字喷泉码的发展历程，并针对几种典型的方案做了详细的描述。LT 码是世界上第一种可行的数字喷泉码方案，它将无码率的特征首次落实到应用中；然后，Raptor 码在 LT 码的基础上进行了扩展，

通过线性预编码的级联，降低了运算复杂度，成为一种新的实用技术；进一步地，A. Shokrollahi 在此基础上设计出了系统的 Raptor 码，使其既保持了原有方案的性能优势，又具备了运算上的简便性。此外，由于预编码自身的特性，使 Raptor 码还具备一般有噪信道下的纠错能力，这也使其应用范围进一步扩大。

- 讨论了数字喷泉码的研究现状，并总结了当前存在的几个难题，包括环的问题、*N* 的控制、译码“死锁”、优先级保护，以及度分布优化等。另一方面，我们也归纳了一些数字喷泉码的衍生应用，这主要包括分布式场景、认知网络和深空通信等，并指出其特定的优势。
- 在实践上的 LT 编码方案中发现了重复关联问题，并围绕该问题展开了研究与讨论。我们先指出了重复关联的具体表现情况，并阐释出其危害性。然后，通过严格的理论分析，揭示了其产生的根本原因，并推导出基于 RSD 度分布下产生无效符号的具体概率。进一步地，我们设计出了三种可行的改进方案，从根本上杜绝了重复关联的发生。最后，通过仿真，确认了实际中产生重复关联问题的严重性，并从对比的实验结果中，总结了各方案的优缺点。
- 从分布式场景出发，介绍了 DSC 理论，并引述了基于 Raptor 码的实现方案。以此为基础，我们进一步论述了 DOS 在数字喷泉码中的重要性，以及 DSC 场景中 Parity 部分的特殊性（绝对的置信能力），从而提出了将 DOS 转移到 Parity 中的设想，并进行了可行性分析。最后，在不影响原码字性能的前提下，我们设计出一套具体的编码方案来验证了这个设想的正确性与优越性。

5.2 未来的展望

在本文中，我们对数字喷泉码做了比较详尽的论述，并针对 LT 编码中的重复关联问题，以及分布式场景中的 Raptor 码方案进行了重点研究。不过，从前面的概述部分可以看出，本文所提及的内容还只是整个数字喷泉码研究领域的“冰山一角”，应该说，该领域还存在进一步发掘的巨大价值。

首先，就本文中的几个研究点来看，也还有进一步深入的意义。对于重复关联问题来说，我们只是阐明了其危害和严重性，对于具体改进的方案，这里只提出了可行的算法，而并没有给出最优化的证明。在分布式场景中，文献[39]中提出了一种分布式联合信源信道编码的设计思想，可以考虑以 DSC 为基础，利用 Raptor 码对其进行尝试。此外，在具体的仿真中，我们并未对所有涉及到的参数

或影响因子做修改和充分的比较，其中可能还存在一些值得发掘的东西。

另一方面，就数字喷泉码的研究现状来说，对于数据的优先级保护、译码“死锁”、度分布优化等问题，目前还有不少工作者在进行相关的探索。这里的任何一个方面，都具有非常大的研究意义，如果能从理论上（或者实践中）进一步提出有效的设计方案，那必将带来很可观应用价值。

此外，数字喷泉码还受到当前一些热门应用的青睐，包括 3G、互联网、认知网络、深空通信、视频、多视角等。这些高速发展的领域与 DFC 技术相结合，必将相互助长、相互促进，其未来的前景也非常可观。

总的来说，数字喷泉码是一项很有前途的纠错技术，其应用前景广泛，具有很高的研究价值。我们也相信，在未来通信中，它必将扮演非常重要的角色。

参考文献

[1] Qusay H. Mahmoud. “Cognitive Networks Towards Self-Aware Networks”. Wiley-Interscience, 2007. pp. 315-331.

[2] 姜博, 晏坚, 蒋卫东. “喷泉码及其在通信网络中的应用”. 数字通信世界, 2007 年第 10 期. pp. 64-67.

[3] http://www.digitalfountain.com/, 2008.

[4] Digital Fountain, Inc. “Information Additive Code Generator and Decoder for Communication Systems”. US Patent 6307487. Oct. 23, 2001.

[5] Digital Fountain, Inc. “Multi-Stage Code Generator and Decoder for Communication Systems”. US Patent 7068729. Jun. 27, 2006.

[6] Digital Fountain, Inc. “Systematic Encoding and Decoding of Chain Reaction Codes”. US Patent 6909383. Jun. 21, 2005.

[7] Digital Fountain, Inc. “Information Additive Group Code Generator and Decoder for Communication Systems”. US Patent 6320520. Nov. 20, 2001.

[8] Digital Fountain, Inc. “Error-Correcting Multi-Stage Code Generator and Decoder for Communication Systems Having Single Transmitters or Multiple Transmitters”. US Patent 7139960. Nov. 21, 2006.

[9] 万奇宝. “喷泉码技术及应用介绍”. 中国卫星大会, 2007 年 9 月.

[10] D. Slepian, J.K. Wolf. “Noiseless Coding of Correlated Information Sources”. IEEE Trans. on Information Theory, Vol. 19, No. 4, Jul. 1973. pp. 471-480.

[11] M. Luby. “LT Codes”. Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS), Nov. 2002. pp. 271-280.

[12] A. Shokrollahi. “Raptor Codes”. Information Theory, IEEE Transactions on. Vol. 52, No. 6, Jun. 2006. pp. 2551-2567.

[13] “速龙码（Raptor Code）：传输新典范”. 中国卫星应用大会资料汇编, 2005 年.

[14] D.J.C. MacKay. “Fountain Codes”. IEE Proc. – Comm. Vol. 152, Dec. 2005. pp. 1062-1068.

[15] 袁东风, 张海刚等. “LDPC 码理论与应用”. 人民邮电出版社, 2008.

[16] R. G. Gallager. “Low Density Parity Check Codes”. IRE Transactions on Information Theory, Vol. IT-8, Jan. 1962. pp. 21-28.

[17] 李庆扬, 王能超, 易大义. “数值计算”, 第 4 版, 清华大学出版社、施普林格出版社. 2005 年 8 月. pp. 161-229.

[18] O. Etesami, A. Shokrollahi. “Raptor Codes on Binary Memoryless Symmetric Channels”. IEEE Transactions on Information Theory, Vol. 52, 2006. pp. 2033-2051.

[19] F. R. Kschischang, B. J. Frey, H.-A. Loeiliger. “Factor Graphs and the Sum-Product Algorithm”. IEEE Transactions on Information Theory, Vol. 47, No. 2, Feb. 2001. pp. 498-519.

[20] 李旭, 张钦宇, 李晖. “喷泉码中相关列提取法去短小环”. 遥测遥控, 第 29 卷第 2 期, 2008 年 3 月.

[21] Digital Fountain, Inc. “Systems and Processes for Decoding Chain Reaction

Codes though Inactivation". US Patent 6856263. Feb. 15, 2005.
[22] S. Kim, K. Ko, S.-Y. Chung. "Incremental Gaussian Elimination Decoding of Raptor Codes over BEC". IEEE Communication Letters, Vol. 12, No. 4, Apr. 2008. pp. 307-309.
[23] M. Nekoui, N. Ranjkesh, F. Lahouti. "A Fountain Code Approach towards Priority Encoding Transmission". Information Theory Workshop, Oct. 2006. pp. 52-55.
[24] 朱宏杰. "喷泉码编译码技术与应用研究". [学位论文]. 清华大学, 电子工程系. 2009 年 4 月.
[25] 朱宏鹏, 张更新, 谢智东. "喷泉码中 LT 码的次优度分布". 应用科学学报, 第 27 卷第 1 期, 2009 年 1 月. pp. 6-11.
[26] S. Puducheri, J. Kliewer, T. E. Fuja. "Distributed LT Codes". IEEE International Symp. on Information Theory, Jul. 2006. pp. 987-991.
[27] Fresia, M. and L. Vandendorpe. "Distributed Source Coding Using Raptor Codes". Proc. IEEE Global Telecommunications Conference, 2007. pp. 1587-1591.
[28] M. Fresia, L. Vandendorpe, H. V. Poor. "Distributed Source Coding Using Raptor Codes for Hidden Markov Sources". Data Compression Conference, 2008. pp. 517-517.
[29] 哈尔滨工业大学深圳研究生院. "深空通信中基于低密度奇偶校验码-喷泉码的编译方法". 中国专利 CN101252606A. 2008 年 8 月 27 日.
[30] Chris Harrelson, Lawrence Ip, Wei Wang. "Limited Randomness LT Codes". Proceedings of the 41st Annu. Allerton Conference on Communication, Control, and Computing. 2003.
[31] Z. Xiong, A. D. Liveris, S. Cheng. "Distributed Source Coding for Sensor Networks". IEEE Signal Processing Magazine, Vol. 21, No. 5, Sep. 2004. pp. 80-94.
[32] J. Garcia-Frias, Y. Zhao. "Compression of Correlated Binary Sources Using Turbo Codes". IEEE Communication Letters, Vol. 5, No. 10, 2001. pp. 417-419.
[33] A.D. Liveris, Z. Xiong, C.N. Georghiades. "Compression of Binary Sources with Side Information at the decoder Using LDPC Codes". IEEE Communication Letters, Vol. 6, 2002. pp. 440-442.
[34] 田宝玉. "工程信息论". 北京邮电大学出版社. 2004 年 8 月.
[35] S.S. Pradhan, K. Ramchandran. "Distributed Source Coding Using Syndromes (DISCUS): Design and Construction". IEEE Transactions on Information Theory, Vol. 49, No. 3, Mar. 2003. pp. 626-643.
[36] B. Girod, A. Aaron, S. Rane, et al. "Distributed Video Coding". Proceedings of the IEEE, Vol. 93, No. 1, 2005. pp. 71-83.
[37] R. Puri, K. Ramchandran. "PRISM: A Video Coding Paradigm Based on Motion-Compensated Prediction at the Decoder". IEEE Transactions on Image Processing, Vol. 16, No. 10, Oct. 2007. pp. 2436-2448.
[38] http://www.cs.toronto.edu/~radford/ftp/LDPC-2006-02-08/index.html, 2006.
[39] X. Q. Zhu, Y. Liu, L. Zhang. "Distributed Joint Source-Channel Coding in Wireless Sensor Networks". J. Sensors, Vol. 6, No. 9, 2009. pp. 4901-4917.

致谢

近三年的硕士生活已接近尾声，回顾这段时间以来的点点滴滴，还是有不少可圈可点的地方，我也为在此期间自己所做出的努力与收获而感到自豪。在这即将离开校园的时刻，我想借此机会，向那些曾经帮助过我的人表示衷心的敬意。

首先，非常感激我的导师张琳副教授在研究生期间对我的器重与栽培。除了学习方面的指导和帮助之外，在生活中，张老师也时常与我畅谈理想和未来，并激励着我的一步步成长。可以说，我这段时间以来的种种收获与成就，都是与他的教导分不开的。

非常感谢实验室的刘雨老师，她在理论与学术方面都为我提供了不少有用的建议和启发，使我受益匪浅。是她让我对理论研究有了更好的理解与把握，并能够积极地去处理好个中问题。

同时，也非常感谢实验室的朱旭琪师姐，是她在很多具体的研究工作中给予我帮助，启发我的思路、纠正我的误解。在我的学术小论文发表之前，她热心地为我修改并给出建议。所以，就最后的结果而言，她也是功不可没的。

这里，特别感谢法语鲁汶大学（UCL）的 Maria Fresia 博士，是她不厌其烦地跟我们就分布式数字喷泉码的问题进行了沟通和交流，并给予我们一些不错的指导与建议，使我们有信心在这个方向上去做进一步的探索。

另外，还要感谢我周围的同学、朋友，他们在生活、学习上都给了我不少帮助和支持，使我可以积极、沉着地面对困境，是他们为我营造了一个舒适、和睦的社会环境。

深深感谢我的父母和家人，感激他们多年以来对我的无私关怀，让我有信心去正视困难、战胜困难、永不放弃，并顺利地一路走来。

最后，再次感谢所有关心、支持和帮助过我的人！

攻读硕士期间发表的学术论文

[1] Zhou Yang, Zhang Lin, Liu Yu. “An Approach of Eliminating Duplicate Associates in Fountain Codes”. Wireless Communications, Networking and Mobile Computing, 2009. Sep. 2009. 第一作者, EI 检索.

[2] Zhou Yang, Zhang Lin, Liu Yu. “An Approach of Eliminating Duplicate Neighbors in Fountain Codes”. The 2nd Joint Workshop between HYU and BUPT, Aug. 2009. 第一作者.