基于陷阱检测的咬尾卷积码译码算法

王晓涛*①②③ 钱 骅②④ 徐 景②④ 杨 旸①
①(上海无线通信研究中心 上海 200335)
②(中国科学院上海微系统与信息技术研究所 上海 200050)
③(中国科学院研究生院 北京 100049)
④(中国科学院无线传感网与通信重点实验室 上海 200335)

摘 要：该文分析了循环维特比算法(CVA)中存在的循环陷阱问题，并证明了传统基于 CVA 的咬尾卷积码译码算法中存在的不足，提出了一种高效率的咬尾卷积码译码算法。该算法通过检测两次不同迭代中获得的两条最大似然路径是否相同来判断是否有循环陷阱产生，并及时终止循环，减少冗余迭代；在没有循环陷阱产生的情况下，新算法比较当前迭代中最大似然路径和已经发现的最优咬尾路径是否相同来自适应终止迭代。文中对循环陷阱检测方案和自适应终止方案做了进一步优化，即利用路径的净增量而非路径本身作为检测量。实验结果表明新算法提高了译码效率，降低了译码复杂度。

关键词：咬尾卷积码；循环维特比算法；循环陷阱；最大似然路径
中图分类号：TN92 **文献标识码**： A **文章编号**：1009-5896(2011)10-2300-06
DOI: 10.3724/SP.J.1146.2011.00413

Trap Detection Based Decoding Algorithm for Tail-biting Convolutional Codes

Wang Xiao-tao①②③ Qian Hua②④ Xu Jing②④ Yang Yang①
①(*Shanghai Research Center for Wireless Communications*, *Shanghai* 200335, *China*)
②(*Shanghai Institute of Microsystem and Information Technology*, *Chinese Academy of Sciences*, *Shanghai* 200050, *China*)
③(*Graduate University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)
④(*Key Laboratory of Wireless Sensor Network & Communication*, *Chinese Academy of Sciences*, *Shanghai* 200335, *China*)

Abstract: There exists circular trap in Circular Viterbi Algorithm (CVA) and deficiencies in CVA-based decoding algorithms of Tail-Biting Convolutional Codes (TBCC). A high efficient decoding algorithm is proposed for TBCC. The checking rule for circular trap in the new algorithm is that comparing whether the two maximum likelihood paths obtained from two different iterations are identical to each other, if they are identical, the CVA should be terminated. Meanwhile, when there no trap happens, a new adaptive stopping rule for CVA is proposed which is based on comparing the maximum likelihood path with the best maximum likelihood tail-biting path. Furthermore, the path used as the measurements in the checking rule and in the stopping rule is replaced by its net path metric to reduce the complexity of decoder. The results of experiments show that the new algorithm improves the decoding efficiency and reduces the decoder complexity.

Key words: Tail-Biting Convolutional Codes (TBCC); Circular Viterbi Algorithm (CVA); Circular trap; Maximum likelihood path

1 引言

卷积码根据其编码器初始化方式的不同，可分为传统卷积码和咬尾卷积码(Tail-Biting Convolutional Codes, TBCC)。传统卷积码的编码器用已知比特(一般是全 0 比特)初始化，并在编码结束的时候使其结束于某个已知状态[1]；TBCC 的编码器用信息序列的最后 v' 位来初始化，其中 $v' \le v$, v 是编码器中的寄存器长度[2,3]。根据 v' 和 v 的关系，TBCC 可分为全咬尾卷积码($v' = v$)和部分咬尾卷积码($v' < v$)，本文只讨论全咬尾卷积码。

2011-05-02 收到，2011-06-10 改回
国家 863 计划项目(2009AA011501)，上海市国际合作项目(10220712100)，上海市重点项目(10511500404)和上海市启明星人才计划(10QA1406300)资助课题
*通信作者：王晓涛 xiaotao.wang@shrcwc.org

采用咬尾方式编码可以消除用已知比特初始化编码器所导致的码率损失，所以 TBCC 被用在多种通信系统中，比如增强型数据速率 GSM 演进技术(Enhanced Data rate for GSM Evolution, EDGE)[4]，微波存取全球互通(worldwide interoperability for microwave access, WiMax)[5]和 3GPP 长期演进项目(Long Term Evolution, LTE)[6]。此外，咬尾的编码方式更适用于分组传输系统[7]。

由于 TBCC 使用信息比特来初始化编码器，这使得它的译码器设计要比传统卷积码复杂。TBCC 的译码器没有关于编码器起始状态的先验信息，所以一种直观的最优译码方案就是对每个可能的起始状态作一次传统维特比译码，最后选取一个最优的结果作为译码输出。这种译码方法即为最大似然译码(Maximum Likelihood Decoding, MLD)。MLD 虽然是最优的，但是复杂度太高。近期，文献[8,9]提出了一种两步骤的 MLD 算法。该算法的第 1 步用修正维特比算法[2]获得对应于每个不同终止状态的幸存路径，并保存所有幸存路径在每个时刻的累积度量值；算法的第 2 步使用最短路径搜索算法找到最优咬尾路径作为译码输出。该算法需要保存大量的中间结果，且两个步骤采用两种算法实现，这对实际应用来说复杂度过高。

循环维特比算法(Circular Viterbi Algorithm, CVA)的提出降低了咬尾卷积码的译码复杂度。CVA 是一种次优译码方案[1,10,11]。文献[10]中提出了可用于 CVA 的两种自适应终止方案，但是由于没有考虑循环陷阱的存在，该算法中存在大量冗余迭代。针对这一问题，本文利用最大似然路径的路径净增量来检测循环陷阱。另外，文中揭示了 CVA 的两个性质。基于这两个性质和对循环陷阱的检测，本文提出了一种新的基于 CVA 的译码算法：陷阱检测-循环维特比译码算法(Trap-Detection CVA, TD-CVA)。

本文结构如下：第 2 节介绍 CVA 的基本概念，给出算法描述中的变量定义，举例说明循环陷阱问题，揭示并证明 CVA 的两个性质；第 3 节结合 CVA 的两个性质和对循环陷阱的检测，提出新的译码算法；第 4 节是实验结果对比分析；最后是结束语。

2　CVA 及其性质

2.1 变量定义与 CVA 算法描述

图 1 中所示是码率为 1/2，约束长度为 3，生成多项式为{7,5}(8 进制表示)的 TBCC 的咬尾格形图。在图 1 中，路径按逆时针方向行走，如箭头所示。从 00 状态起始并结束于 00 状态的路径 P_0 为咬尾路径(用短划线表示)；从 10 状态开始并结束于 01 状态的路径 P_1 为非咬尾路径(用虚线表示)。

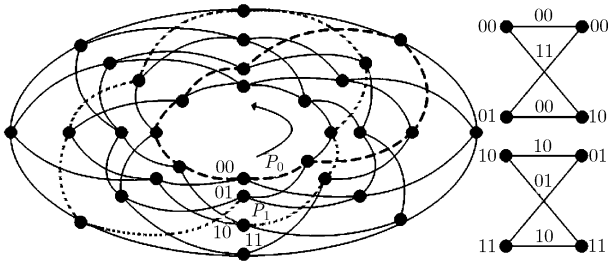

图 1　L=8 的 TBCC 咬尾格形图示例

下面给出文中变量的定义以及 CVA 的算法描述。在咬尾格形图的位置 k 处，有 2^v 个状态，其中 $0 \le k \le L-1$，L 是格形图的长度(即信息序列的长度)。令 S_k 为位置 k 处的状态空间。给定接收序列，每执行一次修正的维特比算法，都能得到一条最大似然路径 P_{ML}。有时 P_{ML} 并不是最优的咬尾路径，这时需要重复执行修正的维特比算法，直到某种终止条件被满足。我们称这样的译码过程为 CVA。

在 CVA 中，状态和路径的累积度量值是从第 1 次迭代开始累积计算的。第 i 次迭代中，$M_k^i(s_k)$ 表示位置 k 处经过状态 s_k 的幸存路径的累积度量值，同时它也是状态 s_k 在第 i 次迭代中位置 k 处的累积度量值；$\beta^i(s_L^\dagger)$ 表示结束于状态 $s_L^\dagger$ 的幸存路径 $P^i(\beta^i(s^\dagger), s_L^\dagger)$ 的起始状态，其中 $s_L^\dagger \in S_L$。幸存路径 $P^i(\beta^i(s_L^\dagger), s_L^\dagger)$ 的路径净增量是它在本次迭代中所经过的分支度量值之和，记为 $M_{\mathrm{net}}^i(\beta^i(s_L^\dagger), s_L^\dagger)$，即 $M_{\mathrm{net}}^i(\beta^i(s_L^\dagger), s_L^\dagger) = M_L^i(s_L^\dagger) - M_0^i(\beta^i(s_L^\dagger))$。相应地，本次迭代中状态 $s_L^\dagger$ 的净增量为 $M_L^i(s_L^\dagger) - M_0^i(s_L^\dagger)$。

每次迭代 CVA 都会找出一条最大似然路径 $P_{\mathrm{ML}}^i(\beta^i(s_L), s_L)$ 和一条最大似然咬尾路径 $P_{\mathrm{MLTB}}^i(s_L^{'}, s_L^{'})$(如果咬尾路径存在)。$P_{\mathrm{ML}}^i(\beta^i(s_L), s_L)$ 和 $P_{\mathrm{MLTB}}^i(s_L^{'}, s_L^{'})$ 的路径净增量分别为 $M_{\mathrm{ML,net}}^i$ 和 $M_{\mathrm{MLTB,net}}^i$。记录前 i 次迭代中找到的路径净增量最大的幸存路径及其路径净增量为(P_{ML}^R，M_{ML}^R)，路径净增量最大的咬尾路径及其路径净增量为(P_{MLTB}^R，M_{MLTB}^R)。设 CVA 允许的最大迭代次数为 N。

2.2 CVA 中的循环陷阱

在 CVA 的译码过程中，若当前循环得到的幸存路径和之前循环中得到的幸存路径相同，则更多的迭代并不能得到路径净增量更大的幸存路径，我们称这种现象为循环陷阱。下面以图 1 中的咬尾格形图为例说明循环陷阱。

幸存路径可以表示为该路径所经过的状态集

合。为表示方便，此处用8进制形式表示状态，比如图中位置 k 处的状态10，记为 $s_k(2)$，其中 $0 \le k \le 7$。对于信息序列{01100000}，编码以后的码字为{(00), (11), (01), (01), (11), (00), (00), (00)}。该码字经过信噪比 $E_b / N_0 = 0$ dB 的加性高斯白噪声(Additive White Gaussian Noise, AWGN)信道，一次实验的接收序列为{(0.490 0.377), (−0.599 −0.774), (−0.292 0.094), (1.051 −1.157), (−0.981 −1.497), (0.926 0.184), (−0.234 0.342), (1.209 −0.952)}。用CVA对上述接收序列译码，译码的过程为：

(1)第 1 次迭代得到的 4 条幸存路径为 P_0^1，P_1^1，P_2^1，P_3^1，其中 P_0^1={ $s_0(0)$，$s_1(0)$，$s_2(2)$，$s_3(3)$，$s_4(1)$，$s_5(0)$，$s_6(0)$，$s_7(0)$，$s_8(0)$ }；P_1^1={ $s_0(0)$，$s_1(0)$，$s_2(2)$，$s_3(1)$，$s_4(0)$，$s_5(2)$，$s_6(3)$，$s_7(3)$，$s_8(1)$ }；P_2^1={ $s_0(0)$，$s_1(0)$，$s_2(2)$，$s_3(3)$，$s_4(1)$，$s_5(0)$，$s_6(0)$，$s_7(0)$，$s_8(2)$ }；P_3^1={ $s_0(0)$，$s_1(0)$，$s_2(2)$，$s_3(3)$，$s_4(1)$，$s_5(0)$，$s_6(0)$，$s_7(2)$，$s_8(3)$ }。每条路径的路径净增量分别为{8.015, 8.689, 7.501, 9.703}。其中最大似然路径 $P_{\mathrm{ML}}^1 = P_3^1$，它的路径净增量为 $M_{\mathrm{ML,net}}^1 = 9.703$；最大似然咬尾路径 $P_{\mathrm{MLTB}}^1 = P_0^1$，它的路径净增量为 $M_{\mathrm{MLTB,net}}^1 = 8.015$。

(2)第 2 次迭代得到的 4 条幸存路径为 P_0^2，P_1^2，P_2^2，P_3^2，其中 P_0^2={ $s_0(1)$，$s_1(2)$，$s_2(3)$，$s_3(3)$，$s_4(1)$，$s_5(0)$，$s_6(0)$，$s_7(0)$，$s_8(0)$ }；P_1^2={ $s_0(3)$，$s_1(1)$，$s_2(0)$，$s_3(0)$，$s_4(0)$，$s_5(2)$，$s_6(3)$，$s_7(3)$，$s_8(1)$ }；P_2^2={ $s_0(3)$，$s_1(1)$，$s_2(0)$，$s_3(2)$，$s_4(3)$，$s_5(1)$，$s_6(2)$，$s_7(1)$，$s_8(2)$ }；P_3^2={ $s_0(1)$，$s_1(2)$，$s_2(3)$，$s_3(3)$，$s_4(1)$，$s_5(0)$，$s_6(0)$，$s_7(2)$，$s_8(3)$ }。每条路径的路径净增量分别为{7.589, 7.139, 6.351, 9.277}。其中最大似然路径为 $P_{\mathrm{ML}}^2 = P_3^2$，它的路径净增量为 $M_{\mathrm{ML,net}}^2 = 9.277$；由于本次迭代没有得到咬尾路径，所以 P_{MLTB}^2 和 $M_{\mathrm{MLTB,net}}^2$ 都不存在。

(3)第 3 次迭代的结果和(2)中的结果相同。

(4)…

从上面的译码过程可以发现，从第 3 次迭代开始幸存路径及其对应的路径净增量出现了循环，这就是循环陷阱。虽然第 1 次迭代中得到的咬尾路径输出就是正确的发送码字，但是传统的译码算法在这里并不能处理循环陷阱，迭代会一直持续下去，直到达到最大允许迭代次数[2,10,11]。以文献[10]中的环绕维特比算法(Wrap-Around Viterbi Algorithm, WAVA)为例，作者提出了两种自适应的终止条件，即简单终止条件和充分终止条件。这两个条件都不能处理上例中的循环陷阱。首先，$P_{\mathrm{ML}}^R = P_{\mathrm{ML}}^1 = P_3^1$，$P_{\mathrm{MLTB}}^R = P_{\mathrm{MLTB}}^1 = P_0^1$，对应的路径净增量分别为 $M_{\mathrm{ML}}^R = 9.703$，$M_{\mathrm{MLTB}}^R = 8.015$。因为 $P_{\mathrm{ML}}^R \ne P_{\mathrm{MLTB}}^R$ 所以简单终止条件失效。其次，从第 2 次迭代开始，各个状态的状态净增量分别为{8.015, 8.153, 7.501, 8.263}。由于 $M_{\mathrm{MLTB}}^R < \min\{8.015, 8.153, 7.501, 8.263\}$，所以充分终止条件失效。

2.3 CVA 的性质

在 2.2 节的例子中，当 $i > 1$ 时，$M_{\mathrm{ML}}^i = 9.277$。可以发现后来迭代中找到的最大似然路径的路径净增量都不大于路径 P_{ML}^1 的路径净增量 $M_{\mathrm{ML,net}}^1 = 9.703$。下面以定理的形式给出 CVA 的这一性质。

定理 1 在 CVA 译码过程中，第 1 次迭代得到的最大似然路径的路径净增量最大，即 $M_{\mathrm{ML,net}}^1 \ge M_{\mathrm{ML,net}}^i$，其中 $i > 1$。

证明 设第 1 次迭代得到的最大似然路径为 $P_{\mathrm{ML}}^1(\beta^1(s), s)$。该路径的累积度量值为 $M_{\mathrm{ML}}^1(\beta^1(s), s)$。由于在 CVA 译码开始的时候，各个起始状态的度量值初始化为 0，所以路径 $P_{\mathrm{ML}}^1(\beta^1(s), s)$ 的路径净增量为 $M_{\mathrm{ML,net}}^1(\beta^1(s), s) = M_{\mathrm{ML}}^1(\beta^1(s), s)$。在第 i 次迭代中，得到的幸存路径 $P^i(\beta^i(s'), s')$ 的路径净增量为

$$M_{\mathrm{net}}^i(\beta^i(s'), s') = M_L^i(s') - M_0^i(\beta^i(s')) \tag{1}$$

在第 i 次迭代中获得的最大似然路径为 $P_{\mathrm{ML}}^i(\beta^i(s^\dagger), s^\dagger)$，它的路径净增量为 $M_{\mathrm{ML,net}}^i(\beta^i(s^\dagger), s^\dagger)$，注意，$s, s', s^\dagger \in S_L$。根据最大似然路径的定义，可以得到

$$M_{\mathrm{net}}^i(\beta^i(s'), s') \le M_{\mathrm{ML,net}}^i(\beta^i(s^\dagger), s^\dagger) \tag{2}$$

下面用反证法来证明定理 1。假设当 $i > 1$ 时有下式成立：

$$M_{\mathrm{ML,net}}^i(\beta^i(s^\dagger), s^\dagger) > M_{\mathrm{ML}}^i(\beta^i(s), s) \tag{3}$$

由最大似然路径的定义，我们知道在第 1 次迭代中有

$$M_{\mathrm{ML}}^1(\beta^1(s), s) \ge M_{\mathrm{net}}^1(\beta^1(s^\dagger), s^\dagger) \tag{4}$$

结合式(3)和式(4)可以得到

$$M_{\mathrm{ML,net}}^i(\beta^i(s^\dagger), s^\dagger) > M_{\mathrm{net}}^1(\beta^1(s^\dagger), s^\dagger) \tag{5}$$

由不等式(5)知道，结束于状态 $s^\dagger$ 的两条路径 $P^i(\beta^i(s^\dagger), s^\dagger)$ 和 $P^1(\beta^1(s^\dagger), s^\dagger)$，前者的路径净增量更大。根据文献[2]中的结论，在第 1 次迭代中得到的结束于状态 $s^\dagger$ 的幸存路径 $P^1(\beta^1(s^\dagger), s^\dagger)$ 的度量值是所有结束于状态 $s^\dagger$ 的路径中路径净增量最大的一条，因此可以得到

$$M_{\mathrm{ML,net}}^i(\beta^i(s^\dagger), s^\dagger) \le M_{\mathrm{net}}^1(\beta^1(s^\dagger), s^\dagger) \le M_{\mathrm{ML}}^1(\beta^1(s), s) \tag{6}$$

不等式(6)与不等式(3)矛盾，因此式(3)中的假设不成立。且式(6)对 $\forall i \in \mathbb{Z}^+$ 成立，由此可得在所有幸存路径中，第 1 次迭代得到的最大似然路径的路径净增量是最大的。 证毕

定理 1 的内容说明了传统算法中用于更新 P_{ML}^{R} 的步骤是多余的，因为 $P_{\mathrm{ML}}^{R}=P_{\mathrm{ML}}^{1}$，所以在 TD-CVA 中删去了这一步骤。

由前面给出的循环陷阱的例子，同样可以发现若第 1 次迭代中找到的 P_{ML}^{1} 和 P_{MLTB}^{1} 不相同，则路径 P_{ML}^{R} 和路径 P_{MLTB}^{R} 在后面的迭代中也不会相同。我们以定理 2 的形式给出上述结论。

定理 2　对 CVA 来说，方程

$$P_{\mathrm{ML}}^{R}=P_{\mathrm{MLTB}}^{R} \tag{7}$$

成立的条件是当且仅当方程

$$P_{\mathrm{ML}}^{1}=P_{\mathrm{MLTB}}^{1} \tag{8}$$

成立。

证明　如果方程 $P_{\mathrm{ML}}^{1}=P_{\mathrm{MLTB}}^{1}$ 成立，则说明第 1 次迭代得到的最大似然路径和最大似然咬尾路径是相同的。CVA 译码过程停止，译码成功，故方程(7)成立。

接下来将证明若方程 $P_{\mathrm{ML}}^{R}=P_{\mathrm{MLTB}}^{R}$ 成立，则方程 $P_{\mathrm{ML}}^{1}=P_{\mathrm{MLTB}}^{1}$ 也成立。因为 P_{ML}^{R} 中存储着到当前迭代为止找到的路径净增量最大的幸存路径，由定理 1 可得

$$P_{\mathrm{ML}}^{R}=P_{\mathrm{ML}}^{1} \tag{9}$$

因此，方程式(7)和方程式：

$$P_{\mathrm{ML}}^{1}=P_{\mathrm{MLTB}}^{R} \tag{10}$$

等价。方程式(10)说明第 1 次迭代得到的最大似然路径和前 i 次迭代中找到的度量值最大的咬尾路径是一样的，即 P_{ML}^{1} 是咬尾路径。因为 P_{ML}^{1} 有最大的路径净增量，且它是咬尾路径，则第 1 次迭代中找到的最大似然咬尾路径也是 P_{ML}^{1}。于是可得

$$P_{\mathrm{ML}}^{1}=P_{\mathrm{MLTB}}^{1} \tag{11}$$

综上可得，方程 $P_{\mathrm{ML}}^{R}=P_{\mathrm{MLTB}}^{R}$ 成立的充要条件是方程 $P_{\mathrm{ML}}^{1}=P_{\mathrm{MLTB}}^{1}$ 成立。　　证毕

定理 2 的内容说明了传统算法中的终止条件 $P_{\mathrm{ML}}^{R}=P_{\mathrm{MLTB}}^{R}$ 效率不高的原因。基于定理 2，下一节会给出更高效率的自适应终止条件。

3　陷阱检测-循环维特比算法

2.2 节中关于循环陷阱的例子说明当循环陷阱发生的时候，更多的迭代并不会得到路径净增量更大的咬尾路径输出。当循环陷阱发生在连续的两次迭代中时，这两次迭代得到的最大似然路径是相同的，因此有循环陷阱的检测方法如下：

$$P_{\mathrm{ML}}^{i}=P_{\mathrm{ML}}^{i-1} \tag{12}$$

为进一步简化式(12)给出的检测方法，此处用路径的路径净增量来代替路径本身作为检测量，于是有

$$M_{\mathrm{ML,net}}^{i}=M_{\mathrm{ML,net}}^{i-1} \tag{13}$$

这样处理可以大大简化检测的复杂度，且简化后的检测方法对性能的影响是可以忽略的。

式(7)就是文献[10]中提出的简单终止条件，由定理 2 可知该终止条件的效率不高。为进一步提高 CVA 的效率，本文提出以下自适应终止条件：

$$P_{\mathrm{ML}}^{i}=P_{\mathrm{MLTB}}^{R} \tag{14}$$

同理，对条件式(14)做如下简化：

$$M_{\mathrm{ML,net}}^{i}=M_{\mathrm{MLTB}}^{R} \tag{15}$$

将上述循环陷阱检测方法和新的终止条件应用到 CVA 中，可以得到更高效率的 TBCC 译码算法，即 TD-CVA 算法。算法步骤如下：

第 1 步　第 1 次迭代，初始化所有起始状态的累积度量值为 0，即对所有 $s\in S_0$，令 $M_0^i(s)=0$。执行修正的维特比算法，找到最大似然路径及其路径净增量(P_{ML}^{1}, M_{ML}^{1})和最大似然咬尾路径及其路径净增量(P_{MLTB}^{1}, M_{MLTB}^{1})(如果它存在)；

第 2 步　若 $M_{\mathrm{ML}}^{1}=M_{\mathrm{MLTB}}^{1}$ 则停止译码，输出路径 P_{MLTB}^{1} 上的码字作为最大似然译码输出；否则保存 P_{ML}^{1}，且若 $M_{\mathrm{MLTB}}^{1}>0$，分别保存 P_{MLTB}^{1} 和 M_{MLTB}^{1} 到 P_{MLTB}^{R} 和 M_{MLTB}^{R} 中；

第 3 步　第 i 次迭代，$i>1$，用 $i-1$ 次迭代中结束于状态 s 的幸存路径的累积度量值来初始化状态 s 的累积度量值，即令 $M_0^i(s)=M_L^{i-1}(s)$；执行修正的维特比算法，找到最大似然路径的路径净增量 $M_{\mathrm{ML,net}}^{i}$ 和最大似然咬尾路径及其路径净增量(P_{MLTB}^{i}, $M_{\mathrm{MLTB,net}}^{i}$)。如果 $M_{\mathrm{MLTB,net}}^{i}>M_{\mathrm{MLTB}}^{R}$，更新 P_{MLTB}^{R} 为 P_{MLTB}^{i}；

第 4 步　如果 $M_{\mathrm{ML,net}}^{i}=M_{\mathrm{ML,net}}^{i-1}$ 或 $M_{\mathrm{ML,net}}^{i}=M_{\mathrm{MLTB}}^{R}$，则停止译码，执行第 6 步，否则执行第 5 步；

第 5 步　如果 $M_{\mathrm{MLTB,net}}^{i}>M_{\mathrm{MLTB}}^{R}$，令 $M_{\mathrm{MLTB}}^{R}=M_{\mathrm{MLTB}}^{i}$，回到第 3 步，直到达到最大迭代次数 N；

第 6 步　若路径 P_{MLTB}^{R} 存在，则输出该路径对应的码字序列；否则输出路径 P_{ML}^{1} 对应的码字序列。

在 TD-CVA 中，条件 $M_{\mathrm{ML,net}}^{i}=M_{\mathrm{ML,net}}^{i-1}$ 可以及时检测出循环陷阱，从而减少冗余迭代的产生。在 2.2 节的例子中，第 3 次迭代得到的幸存路径和第 2 次迭代得到的一样，于是有 $P_{\mathrm{ML}}^{3}=P_{\mathrm{ML}}^{2}$，同时可检测出 $M_{\mathrm{ML,net}}^{3}=M_{\mathrm{ML,net}}^{2}=9.277$。CVA 在第 3 次迭代结束以后就可以被终止，译码输出为咬尾路径 P_0^1 对应的信息序列{01100000}。

即使没有循环陷阱产生，TD-CVA 也比传统的基于 CVA 的译码算法效率更高。定理 1 说明 P_{ML}^{1} 的路径净增量是所有幸存路径中最大的，因此在 TD-CVA 中删去了传统算法中用于更新 P_{ML}^{R} 的步骤[1,2,10]。传统算法中用条件 $P_{\mathrm{ML}}^{R}=P_{\mathrm{MLTB}}^{R}$ 来终止译

码[10]，定理 2 说明该条件若在第 1 次迭代中不成立，则在后面的迭代中再也不会成立，即使在后面的迭代中找到了正确的咬尾路径，该条件也会失效。而新的终止条件 $P_{\mathrm{ML}}^{i}=P_{\mathrm{MLTB}}^{R}$ 可以更加有效地处理这些情况。下面举例说明新条件的有效性。

此处依然采用图 1 中的生成多项式。设输入的信息序列为{11100100}，编码器生成的码字为{(11), (01), (10), (01), (11), (11), (10), (11)}。在通过信噪比 $E_b/N_0=0$ dB 的 AWGN 信道后(一次实验)，接收到的软信息序列为{(−1.409 −0.283), (0.276 −0.159), (−1.914 −0.561), (0.109 0.179), (−1.208 −0.708), (−1.273 −2.074), (−0.842 0.716), (−0.781 −0.176)}。CVA 第 1 次迭代得到的 4 条幸存路径分别为 $P_0^1=\{s_0(1),\ s_1(0),\ s_2(0),\ s_3(2),\ s_4(1),\ s_5(0),\ s_6(2),\ s_7(1),\ s_8(0)\}$；$P_1^1=\{s_0(3),\ s_1(3),\ s_2(1),\ s_3(0),\ s_4(2),\ s_5(1),\ s_6(0),\ s_7(2),\ s_8(1)\}$；$P_2^1=\{s_0(1),\ s_1(0),\ s_2(0),\ s_3(2),\ s_4(1),\ s_5(0),\ s_6(2),\ s_7(1),\ s_8(2)\}$；$P_3^1=\{s_0(3),\ s_1(3),\ s_2(1),\ s_3(0),\ s_4(0),\ s_5(2),\ s_6(3),\ s_7(3),\ s_8(3)\}$，它们的路径净增量分别为{12.132, 8.326, 10.218, 9.204}。最大似然路径 $P_{\mathrm{ML}}^1=P_0^1$，最大似然咬尾路径 $P_{\mathrm{MLTB}}^1=P_3^1$。因此 $P_{\mathrm{MLTB}}^R=P_{\mathrm{MLTB}}^1=P_3^1$，$M_{\mathrm{MLTB}}^R=M_{\mathrm{MLTB,net}}^1=9.204$。第 2 次迭代得到的 4 条幸存路径分别为 $P_0^2=\{s_0(0),\ s_1(2),\ s_2(3),\ s_3(3),\ s_4(1),\ s_5(0),\ s_6(2),\ s_7(1),\ s_8(0)\}$；$P_1^2=\{s_0(0),\ s_1(2),\ s_2(1),\ s_3(0),\ s_4(2),\ s_5(1),\ s_6(0),\ s_7(2),\ s_8(1)\}$；$P_2^2=\{s_0(0),\ s_1(2),\ s_2(3),\ s_3(3),\ s_4(1),\ s_5(0),\ s_6(2),\ s_7(1),\ s_8(2)\}$；$P_3^2=\{s_0(0),\ s_1(2),\ s_2(1),\ s_3(0),\ s_4(0),\ s_5(2),\ s_6(3),\ s_7(3),\ s_8(3)\}$。它们的路径净增量分别为{11.188, 8.022, 9.274, 8.900}。本次迭代中，最大似然路径同时也是咬尾路径，即 $P_{\mathrm{ML}}^2=P_{\mathrm{MLTB}}^2=P_0^2$，且 $M_{\mathrm{ML,net}}^2=11.188>M_{\mathrm{MLTB}}^R=9.204$，更新 P_{MLTB}^R 为 P_{MLTB}^2，更新 M_{MLTB}^R 为 11.188。此时，终止条件 $M_{\mathrm{ML,net}}^i=M_{\mathrm{MLTB}}^R$ 可以终止 CVA 的循环，并将正确的译码结果输出。条件 $M_{\mathrm{ML,net}}^i=M_{\mathrm{MLTB}}^R$ 和条件 $P_{\mathrm{ML}}^i=P_{\mathrm{MLTB}}^R$ 等效，但是前者却可以更快速地做出判决。而条件 $P_{\mathrm{ML}}^R=P_{\mathrm{MLTB}}^R$ 在此处的例子中也是无效的。

4 实验结果对比分析

将本文中给出的 TD-CVA 算法与文献[10]中的 WAVA 算法进行比较；WAVA 算法中使用简单终止条件时记为 WAVA-Simple，使用充分终止条件时记为 WAVA-Sufficient。简单终止条件即式(7)；充分终止条件的实现需要在每次迭代后更新每个终止状态的状态净增量，并需要将这些值和 M_{MLTB}^R 作比较，算法实现复杂度高。这里比较不同算法的误块率性能和所需的平均迭代次数。仿真条件为：AWGN 信道，码字比特采用 QPSK 调制。仿真中设的最大允许迭代次数为 $N=20$。

第 1 组仿真针对 TBCC 在 LTE 中的应用。LTE 中广播信道使用的卷积码生成多项式为{133, 171, 165}[6]，码率为 1/3，约束长度为 7；输入到编码器的信息序列长度为 40 bit。采用咬尾方式编码可以减少 13%的有效码率损失。图 2 所示为 TD-CVA 和 WAVA 的误块率性能，可见 TD-CVA 算法的译码性能逼近 MLD 的性能。图 3 所示为各种算法所需要的平均迭代次数，可以发现相对于 WAVA 而言，TD-CVA 显著降低了平均迭代次数，极大的提高了译码效率。MLD 的迭代次数为常数 64。TD-CVA 算法在提升译码效率的同时，还降低了译码复杂度和对存储空间的需求；WAVA-Sufficient 虽然相对于 WAVA-Simple 来说也只需要较少的迭代次数，但是 WAVA-Sufficient 的实现复杂度高。

第 2 组仿真针对 TBCC 在 EDGE 中的应用。EDGE 中 Type 5 的分组数据块的数据头采用码率为 1/3 的咬尾卷积码，其生成多项式为{133, 171, 145}，约束长度为 7[4]。送入编码器的数据头的长度为 36 bit，此处不考虑打孔。采用咬尾方式编码可以减少 15%的有效码率损失。TD-CVA 和 WAVA 算法得到的误块率性能如图 4 所示，各种算法需要的平均迭代次数如图 5 所示。可以得到和第 1 组仿真中相似的结论。

5 结束语

传统的基于 CVA 的咬尾卷积码译码算法中存在循环陷阱，循环陷阱会导致译码过程中产生大量冗余迭代。本文针对循环陷阱的问题，提出了一种通过检测最大似然路径是否相同的方法来检测陷阱的产生。同时，文中对 CVA 算法的两个性质进行了阐述并给予了证明。利用这两个性质，本文提出了一种的新的可用于 CVA 算法的自适应迭代终止方案。通过融合循环陷阱检测方案和新的自适应迭代终止方案到 CVA 中，本文提出了 TD-CVA 算法。TD-CVA 算法可以检测出循环陷阱，从而减少冗余迭代，提高迭代效率。此外，文中对提出的循环陷阱检测方案和自适应迭代终止方案进行了优化，通过比较路径净增量而不直接对路径本身进行比较，进一步提高了检测效率，减少了译码时间。由于本文提出的循环陷阱检测方案只能在陷阱发生的时候检测到，所以下一步的工作方向是探索如何更早预测到循环陷阱的产生，从而彻底消除冗余迭代。

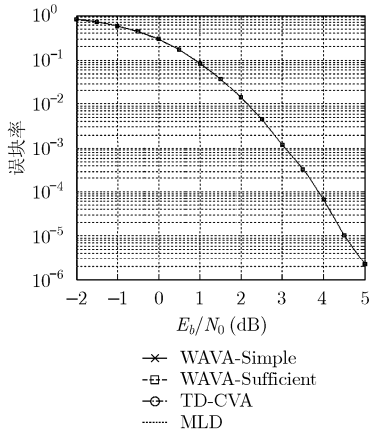

图 2 采用 LTE 中的卷积码，不同算法的误块率性能

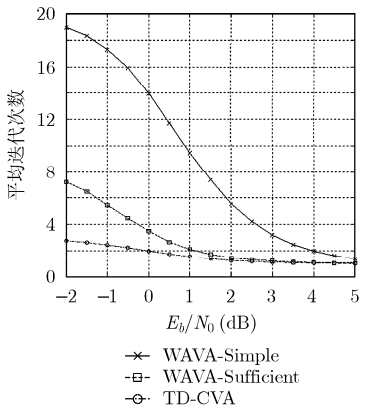

图 3 采用 LTE 中的卷积码，不同算法所需的平均迭代次数

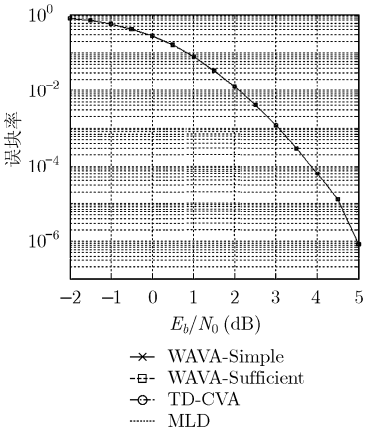

图 4 采用 EDGE 中的卷积码，不同算法的误块率性能

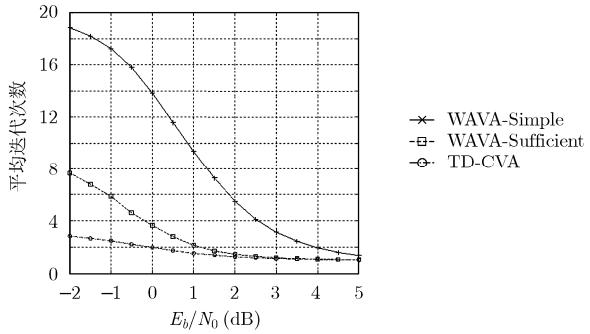

图 5 采用 EDGE 中的卷积码，不同算法所需的平均迭代次数

参 考 文 献

[1] Cox V C and Sundberg C W. An efficient adaptive circular Viterbi algorithm for decoding generalized tailbiting convolutional codes [J]. *IEEE Transactions on Vehicular Technology*, 1994, 43(1): 57–68.

[2] Wang Q and Bhargava V K. An efficient maximum-likelihood decoding algorithm for generalized tailbiting convolutional codes including quasi-cyclic codes [J]. *IEEE Transactions on Communications*, 1989, 37(8): 875–879.

[3] Ma H H and Wolf J K. On tail biting convolutional codes [J]. *IEEE Transactions on Communications*, 1986, 34(2): 104–111.

[4] 3GPP TS. 45.003-3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Channel Coding (Release 9) [S]. 2009.

[5] IEEE 802.16-2009. IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Broadband Wireless Access Systems [S]. 2009.

[6] 3GPP TS. 36.212-3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding (Release 8) [S]. 2009.

[7] Stahl P, Anderson J B, and Johannesson R. Optimal and near-optimal encoders for short and moderate-length tailbiting trellises [J]. *IEEE Transactions on Information Theory*, 1999, 45(7): 2562–2571.

[8] Pai H T, Han S Y, Wu T, *et al.*. Low-complexity ML decoding for convolutional tail-biting codes [J]. *IEEE Communications Letters*, 2008, 12(12): 883–885.

[9] Shankar P, Kumar P N A, Sasidharan K, *et al.* Efficient convergent maximum likelihood decoding on tail-biting. http://arxiv.org/abs/cs.IT/0601023. 2011.4.

[10] Shao R Y, Lin S, and Fossorier M P C. Two decoding algorithms for tailbiting codes [J]. *IEEE Transactions on Communications*, 2003, 51(10): 1658–1665.

[11] Zhang M, Huang J, Meng J, *et al.*. Research on an-based decode of tail-biting convolutional codes and their performance analyses used in LTE system [C]. 2009 International Forum on Information Technology and Applications, Chengdu, China, May 15–17, 2009, 2: 303–306.

王晓涛：男，1985 年生，博士生，研究方向为 LTE 物理层关键技术研究.

钱　骅：男，1976 年生，研究员，博士生导师，研究领域包括通用的无线通信系统物理层关键算法与实现、非线性信号处理、无线传感网络等.

徐　景：男，1975 年生，副研究员，硕士生导师，研究领域主要包括物理层传输、跨层优化、中继技术、B3G/4G 通信系统网络架构设计及资源管理、系统性能分析和相关技术的标准化.

杨　旸：男，1974 年生，研究员，博士生导师，研究领域包括 3G/4G 移动通信网络、无线传感器和 MESH 网络、动态无线资源分配算法、媒质接入控制(MAC)协议、无线多媒体业务流特征分析、跨层性能分析和优化、移动自组织网络.