基于 Viterbi-双向搜索的咬尾码最大似然译码算法

王晓涛[①②③] 钱 骅*[①④] 康 凯[①④]
[①](中国科学院上海微系统与信息技术研究所 上海 200050)
[②](上海无线通信研究中心 上海 200335)
[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：传统咬尾码最大似然(ML)译码算法在译码时存在两个问题：复杂度高和消耗存储空间大。针对这两个问题，该文提出了一种基于 Viterbi 算法和双向搜索算法的最大似然译码算法。新算法利用 Viterbi 算法得到的幸存路径度量值与最大似然咬尾路径度量值的关系，删除不可能的起始状态及其对应的咬尾格形子图，缩小搜索空间；然后利用双向搜索算法中门限值与最大似然咬尾路径度量值的关系来降低双向搜索算法的复杂度，从而得到一种在咬尾格形图上高效率的最大似然译码算法。新的最大似然译码算法不仅降低了译码复杂度，同时降低了译码器对存储空间的需求。

关键词：编码；咬尾码；咬尾格形图；最大似然译码；双向搜索算法
中图分类号：TN92 **文献标识码**： A **文章编号**：1009-5896(2013)05-1017-06
DOI: 10.3724/SP.J.1146.2012.01219

Viterbi-bidirectional Searching Based ML Decoding Algorithm for Tail-biting Codes

Wang Xiao-tao[①②③] Qian Hua[①④] Kang Kai[①④]
[①](*Shanghai Institute of Microsystem and Information Technology*, *Chinese Academy of Sciences*, *Shanghai* 200050, *China*)
[②](*Shanghai Research Center for Wireless Communications*, *Shanghai* 200335, *China*)
[③](*Graduate University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)
[④](*Key Laboratory of Wireless Sensor Network & Communication*, *Shanghai Institute of Microsystem and Information Technology*, *Chinese Academy of Sciences*, *Shanghai* 200335, *China*)

Abstract: There exist two problems with the conventional Maximal Likelihood (ML) decoding algorithms: high decoding complexity and large memory space consumption. To solve these problems, a new algorithm that is based on Viterbi and bidirectional searching algorithm is proposed. By comparing the accumulated path metrics of survived paths with the path metric of ML tail-biting path, all of which are obtained in the Viterbi searching phase, the new algorithm deletes impossible starting states and their corresponding sub-tail-biting trellises to reduce the searching space for the second phase. In the second phase, the decoding complexity can be further reduced by comparing the path metric of ML tail-biting path with the threshold used in the bidirectional searching algorithm. Combing the Viterbi algorithm and bidirectional searching algorithm, a new ML decoding algorithm for tail-biting codes, which can be performed on tail-biting trellis with high efficiency, is obtained. The results of experiments show that the new algorithm improves the decoding efficiency and reduces the memory space consumption.

Key words: Coding; Tail-biting codes; Tail-biting trellis; Maximal Likelihood (ML) decoding; Bidirectional searching algorithm

1 引言

咬尾卷积编码是一种非常有效的卷积码编码方式，它能够消除用已知比特来初始化编码器所导致的码率损失，而这种码率损失在信息序列较短时是不容忽视的[1−3]。正是由于咬尾卷积码的这种优势，使得它被广泛运用于各种无线通信系统的控制信道和广播信道，如增强型数据速率 GSM 演进技术(Enhanced Data rate for GSM Evolution, EDGE)和 3GPP 长期演进项目(Long Term Evolution,

2012-09-19 收到，2012-11-02 改回
中国科学院“百人计划”，上海市浦江人才计划(11PJ1408700)，国家科技重大专项(2011ZX03003-003-04)和科技部国际科技合作项目(2012DFG12060)资助课题
*通信作者：钱骅 hua.qian@shrcwc.org

LTE)[4,5]。采用传统的卷积编码所得到的码字可以用传统格形图来表示，传统格形图只有一个起始状态和一个终止状态，而采用咬尾方式编码得到的码字可以用咬尾格形图来表示[6]。咬尾格形图含有多个起始状态和对应的多个终止状态，且首尾相连[7,8]。

在文献[9]中，作者通过研究发现准循环码和咬尾卷积码间具有一一对应关系，因此准循环码也可以用咬尾格形图来表示。近些年的研究表明，许多分组码也可以用咬尾格形图表示[6−8]。而且对同样的分组码来说，它的咬尾格形图表示比传统格形图表示具有更简洁的形式[6]。比如对(24,12)Golay 码来说其传统格形图上中间位置的状态数高达 512[10]，而其咬尾格形图上的状态数仅为 16[6]。本文中将咬尾卷积码和可以用咬尾格形图表示的分组码定义为咬尾码。

对基于格形图的译码算法进行研究，不仅对研究码字的结构具有重要意义，同时对研究咬尾码的软输入最大似然译码算法也有重要意义。目前基于咬尾格形图的译码算法主要有基于 Viterbi-启发式搜索(Viterbi-Heuristic, VH)算法的最大似然译码算法[11,12]和基于循环 Viterbi 算法的次优译码算法，如 TD-CVA[2]。其中 VH 算法对存储空间的要求很高，这是因为在 VH 算法执行过程中，所有 Viterbi 算法搜索的幸存路径在每个时刻的度量值都要保存起来，用于启发式搜索过程中启发式函数 f 值的估计。双向搜索算法也是咬尾码的一种高效最大似然译码算法，不过它只能工作在传统格形图上，即格形图上只有一个起始状态和一个终止状态[10]。在咬尾格形图上，双向搜索算法需要从每个可能的起始状态开始做一次双向搜索，穷尽搜索各个起始状态的咬尾格形子图，这导致译码复杂度过高。

本文借鉴双向搜索算法中的门限值思想，设计一种基于 Viterbi 算法和双向搜索算法的两步骤最大似然译码算法。新的译码算法将利用 Viterbi 算法中获得的幸存路径度量值来设置双向搜索算法中的门限值。利用这些信息可以更加准确地预测各个咬尾格形子图上存在的咬尾路径的度量值门限，从而提高双向搜索算法在咬尾格形图上的搜索效率，降低译码复杂度的同时减少对存储空间的需求。

本文结构如下：第 2 节介绍新算法的设计原理，并对算法的最优性进行证明；第 3 节给出算法性能仿真验证；第 4 节总结全文。

2 算法描述

2.1 变量定义

如图 1 所示的咬尾格形图 T 是生成多项式为{7,5}的咬尾卷积编码器生成，其中实线连接各状态组成的是状态 00 的咬尾格形子图 $T_{\text{sub}}(00)$，它是由咬尾格形图上所有从 00 状态起始并结束到 00 状态的咬尾路径组成。图中粗实现连接的状态构成的是 $T_{\text{sub}}(00)$ 上的一条咬尾路径。

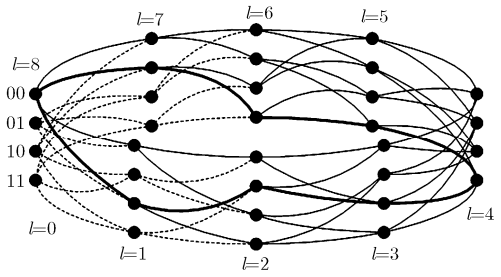

图 1 生成多项式为{7,5}的咬尾卷积码格形图

咬尾格形图长度为 L (=8)，在每个时刻 l 处的状态空间定义为 S_l，其中 $0 \le l \le L$。对咬尾格形图来说有 $S_0 = S_L$。在图 1 所示的咬尾格形图上，每个时刻的状态空间是相同的(并不是所有的咬尾格形图都具有此特性)，即 $S_l = \{00, 01, 10, 11\}$。

本文所提算法分为两个步骤，第 1 步是基于 Viterbi 算法。在咬尾格形图上，Viterbi 算法从位置 $l = 0$ 处的所有起始状态开始搜索，搜索到 $l = L$ 处，会得到 $|S_L|$ 条幸存路径，其中 $|S_L|$ 表示集合 S_L 中的状态个数。幸存路径 $P(s',s)$ 起始于时刻 $l = 0$ 处的状态 s' 并结束于时刻 $l = L$ 的状态 s，即 $s' \in S_0$，$s \in S_L$。路径 $P(s',s)$ 的累积度量值记为 $M_L(s)$。若路径 $P(s',s)$ 的起始状态和终止状态相同，即若 $s = s'$，则该路径为咬尾路径。对于咬尾码来说，其最大似然译码等效于在咬尾格形图上寻找最大似然咬尾路径所对应的码字，相应的记最大似然咬尾路径及其度量值为 P_{MLTB} 和 M_{MLTB}。

下面以码率为 b/c 的咬尾卷积码为例来阐述幸存路径累积度量值的计算方法。设咬尾码字序列为 $\boldsymbol{v}=\{v_l,\ 0 \le l \le Lc\}$，假设采用 BPSK 调制，则调制以后的符号序列为 $\boldsymbol{x}=\{x_l, 0 \le l \le Lc\}$，其中 $x_l = (1-2v_l)\sqrt{E_s}$。不失一般性将 E_s 归一化为 1。通过噪声功率谱密度为 $N_0/2$ 的加性高斯白噪声(Additive White Gaussian Noise, AWGN)信道以后，对应的接收序列为 $\boldsymbol{r}=\{r_l,\ 0 \le l \le Lc\}$。则 r_l 的对数似然比信息为

$$\text{LLR}(r_j) = \ln \frac{\Pr(r_j \mid x_j = +1)}{\Pr(r_j \mid x_j = -1)} = 4r_j/N_0$$

在 Viterbi 搜索过程中，时刻 l 处的最大似然路径累积度量值为

$$M_l = \arg\max_{\boldsymbol{x}} \sum_{j=0}^{Lc-1} x_j \text{LLR}(r_j) \tag{1}$$

加权汉明距离定义为[10]

$$D(\mathrm{LLR}(r_j),x_j)=\begin{cases}|\,\mathrm{LLR}(r_j)\,|, & \mathrm{sign}(r_j)\neq x_j\\ 0, & \text{其它}\end{cases}$$

其中 $\mathrm{sign}(r_j)$ 表示 r_j 的符号位，这样式(1)可以写为

$$\begin{aligned}M_l&=\arg\max_{\boldsymbol{x}}\sum_{j=0}^{Lc-1}|\mathrm{LLR}(r_j)\,|-2D(\mathrm{LLR}(r_j),x_j)\\&=\arg\min_{\boldsymbol{x}}\sum_{j=0}^{Lc-1}D(\mathrm{LLR}(r_j),x_j)\end{aligned}\tag{2}$$

在式(2)的第 2 行中 $|\mathrm{LLR}(r_j)\,|$ 项被忽略，是因为该项独立于符号序列 $\boldsymbol{x}$，即与具体路径无关。利用式(2), Viterbi 算法演变为在整个咬尾格形图上寻找满足式 $M_L=\arg\min_{\boldsymbol{x}}\sum_{j=0}^{Lc-1}D(\mathrm{LLR}(r_j),x_j)$ 的最大似然路径。然而，最大似然路径并不一定是咬尾路径。当 Viterbi 搜索所得到的最大似然路径不是咬尾路径时，利用双向搜索算法确定全局最优咬尾路径 P_{MLTB}。

2.2 算法描述

下面给出算法的具体步骤，为引用方便，本文所提算法简写为 VBS(Viterbi-Bidirectional Searching, VBS)算法：

初始化：令起始于 S_0 中所有状态的路径累积度量值的初始值为 0，即令 $M_0(s)=0$，$\forall s\in S_0$；$M_{\mathrm{MLTB}}=+\infty$。

Viterbi 搜索：从 S_0 中所有状态开始进行 Viterbi 搜索，得到 $|S_L|(=|S_0|)$ 条幸存路径 $P=\{(P(s',s),M(s)),\ s\in S_L\}$，若 P 中有咬尾路径，则将累积度量值最小的咬尾路径及其度量值保存在 $(P_{\mathrm{MLTB}},M_{\mathrm{MLTB}})$ 中：

情况 1　若 $M_{\mathrm{MLTB}}=\min(\{M(s),\forall s\in S_L\})$，输出路径 P_{MLTB} 上的码字，译码结束；

情况 2　否则将所有累积路径度量值小于 M_{MLTB} 的幸存路径结束状态按度量值的升序保存到集合 C 中，即 $C=\{(s,T_{\mathrm{sub}}(s),M(s))|$ 若 $i<j$，则 $M(s_i)\le M(s_j),\ s_i,s_j\in S_L\}$。

双向搜索：在传统格形图上，通过设定门限值来约束搜索空间，双向搜索算法能够以非常高的效率找到最大似然咬尾路径[10]。在咬尾格形图上，通过 Viterbi 搜索，得到集合 C，这相当于从最初的 $|S_0|$ 个咬尾格形子图中通过筛选得到剩下的 $|C|$ 个待搜索咬尾格形子图。在搜索过程中门限值 Th 不断增大而 M_{MLTB} 不断减小，这时会有更多的状态 s 及其 $T_{\mathrm{sub}}(s)$ 被从 C 中排除，搜索范围进一步缩小。门限值的最大允许值为 $\mathrm{Th}_{\max}=M_{\mathrm{MLTB}}$。当 $\mathrm{Th}\ge M_{\mathrm{MLTB}}$ 时，译码结束，此时双向搜索算法将会收敛到全局最优咬尾路径，且这个路径会被保存在 P_{MLTB} 中。

下面简单阐述基于集合 C 的双向搜索算法译码过程。设集合 C 中相邻的 3 个状态为 s_{i-1},s_i,s_{i+1}，对应的有 $M(s_{i-1})\le M(s_i)\le M(s_{i+1})$。首先在 s_{i-1} 的咬尾格形子图 $T_{\mathrm{sub}}(s_{i-1})$ 上进行搜索，在搜索过程中若发现在 $T_{\mathrm{sub}}(s_{i-1})$ 上的搜索门限值 Th_1 已经超过 $M(s_i)$ 则暂停搜索，并保存中间结果。此时启动在 $T_{\mathrm{sub}}(s_i)$ 上的搜索，直到 s_i 上的门限值也超过 Th_1，此时暂停在 $T_{\mathrm{sub}}(s_i)$ 上的搜索，并保存中间结果。增大门限值到 Th_2，并继续在 $T_{\mathrm{sub}}(s_{i-1})$ 和 $T_{\mathrm{sub}}(s_i)$ 上的搜索，直到 Th_2 大于 $M(s_{i+1})$，暂停在 $T_{\mathrm{sub}}(s_{i-1})$ 和 $T_{\mathrm{sub}}(s_i)$ 上的搜索；继续增大门限值同时启动在 $T_{\mathrm{sub}}(s_{i-1})$，$T_{\mathrm{sub}}(s_i)$ 和 $T_{\mathrm{sub}}(s_{i+1})$ 上的搜索。以此类推，直到 $\mathrm{Th}=\mathrm{Th}_{\max}$。

在搜索过程中：(1)若新发现的咬尾路径累积度量值小于 M_{MLTB}，则更新 $(P_{\mathrm{MLTB}},M_{\mathrm{MLTB}})$；(2)若所设门限值 Th 已不小于 M_{MLTB}，则译码结束。$(P_{\mathrm{MLTB}},M_{\mathrm{MLTB}})$ 中保存的就是最大似然咬尾路径。

下面通过一个例子来说明VBS算法的译码过程。

例 1　LTE中的咬尾卷积编码器生成多项式为 $\{133,171,165\}$[5]。设信息序列为{101000101101}，咬尾编码以后的码字序列为{100 001 101 011 010 101 001 111 111 010 101 010}。为方便，编码器的状态用十进制表示。通过信噪比 $E_b/N_0=0$ dB的AWGN信道，一次实验中得到的接收序列为{(−2.208 1.854 −0.389), (0.271 1.942 −0.690), (−0.206 1.254 −0.366), (0.315 0.327 0.938), (1.039 −0.472 0.799), (−1.984 0.260 −0.208), (−1.034 0.563 −1.921), (−0.062 0.358 0.129), (−0.931 −1.383 −1.936), (1.650 −2.040 0.395), (−0.165 2.555 0.064), (1.686 −0.928 −0.236)。通过Viterbi搜索得到64条幸存路径集合 P，其中咬尾路径为 P_{MLTB} (26,26)，其度量值 M_{MLTB}=3.655。集合 P 中度量值比 M_{MLTB} 小的非咬尾路径有两条，对应的结束状态分别为16和45，路径度量值分别为3.332和2.902，因此 $C=\{(45,T_{\mathrm{sub}}(45),2.902),(16,\ T_{\mathrm{sub}}(16),3.332)\}$，故双向搜索算法需要在 $T_{\mathrm{sub}}(45)$ 和 $T_{\mathrm{sub}}(16)$ 上进行搜索。双向搜索算法在 $T_{\mathrm{sub}}(45)$ 上的译码过程如图2(a)所示，门限值的上限为 $\mathrm{Th}_{\max}=M_{\mathrm{MLTB}}=3.655$。门限值 Th 从0增加到 $\mathrm{Th}_{\max}$，前/后向搜索最大门限为 $\mathrm{Th}_{\max}/2=1.828$。在前/后向搜索过程中，所有度量值大于 Th/2 的路径都将被删除，对应的结束状态也将被删除，这些状态对应图2(a)中的空心圆。前向搜索和后向搜索在 $l=6$ 处的状态5相遇，得到 $T_{\mathrm{sub}}(45)$ 上的最大似然咬尾路径为图2(a)中实心圆连接构成，$(P_{\mathrm{MLTB}}(26,26),M_{\mathrm{MLTB}})$ 更新为 $(P_{\mathrm{MLTB}}(45,45),$

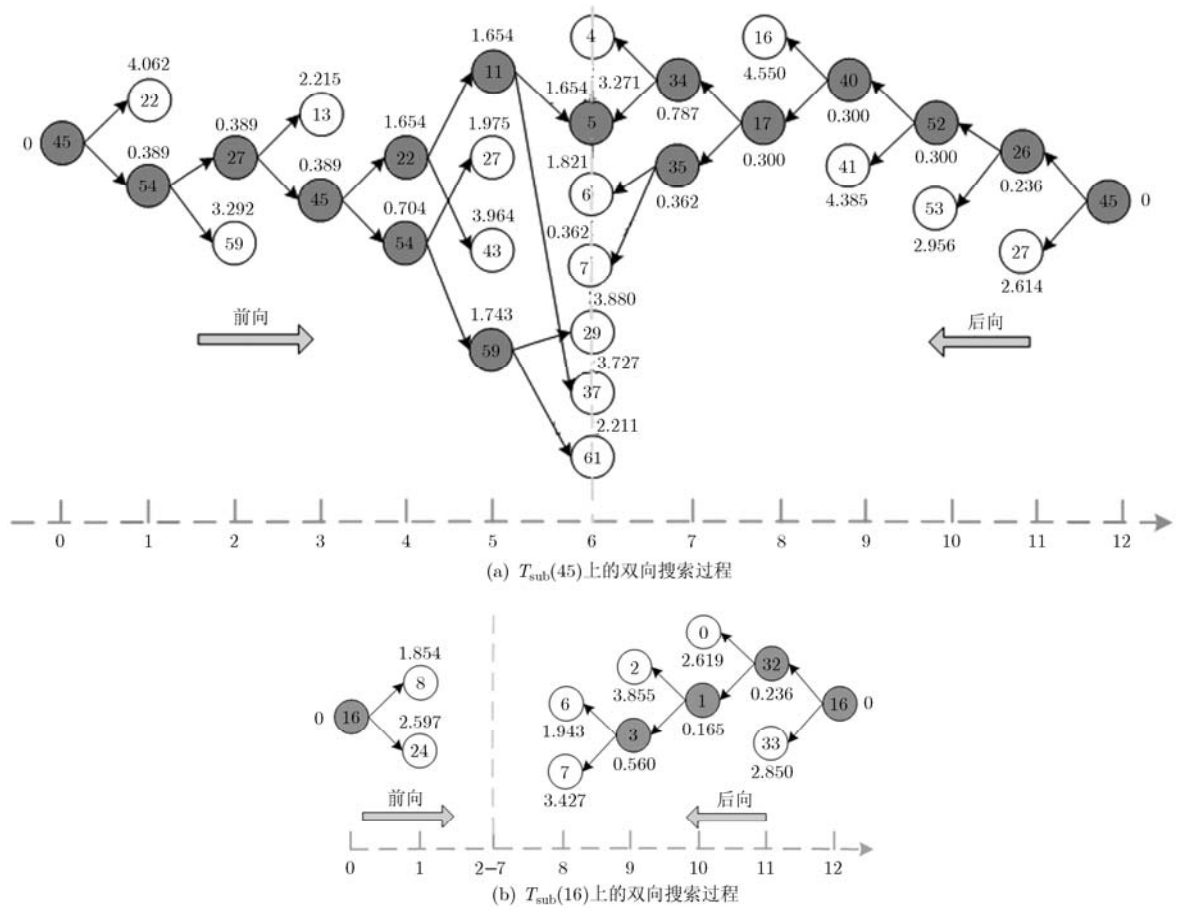

图2 双向搜索过程示意图

3.475)。在$T_{\text{sub}}(16)$上的双向搜索最大门限值更新为$\text{Th}_{\max}=3.475$，前向搜索和后向搜索最大门限为$\text{Th}_{\max}/2=1.738$。由图2(b)可以发现，在$T_{\text{sub}}(16)$上的前向搜索结束在$l=1$处，后向搜索结束在$l=8$处，没有找到咬尾路径。路径$P_{\text{MLTB}}(45,45)$对应的信息序列为{101000101101}，与发送信息序列相同，译码成功。

2.3 最优性证明

定理 1 VBS 算法是咬尾码的最大似然译码算法。

证明 分为 3 种情况讨论

(1)Viterbi 搜索得到的度量值最小的路径为咬尾路径，保存在$(P_{\text{MLTB}},M_{\text{MLTB}})$中；

(2)Viterbi 搜索得到的度量值最小的咬尾路径为$(P_{\text{MLTB}},M_{\text{MLTB}})$，但$P_{\text{MLTB}}$不是所有幸存路径中度量值最小的；

(3)Viterbi 搜索没有找到咬尾路径。

首先证明情况(1)，即证明：若P_{MLTB}是P中度量值最小的路径，则它是全局最优咬尾路径。

在 Viterbi 搜索中得到幸存路径集合为$P=\{(P(s',s),M(s)),s\in S_L\}$。由文献[13]中的定理 1 可知 Viterbi 算法具有这样的性质：在咬尾格形图上所有结束到状态s的可能的路径中，幸存路径$P(s',s)$的度量值$M(s)$是最小的。设状态s的咬尾格形子图$T_{\text{sub}}(s)$上的任一咬尾路径及其路径度量值为$(P_{\text{TB}}(s,s),M_{\text{TB}}(s))$，因此有

$$M_{\text{TB}}(s,s)\geq M(s) \tag{3}$$

式(3)说明$T_{\text{sub}}(s)$上的所有咬尾路径的度量值$M_{\text{TB}}(s,s)$都不小于 Viterbi 搜索所得到的幸存路径$P(s',s)$的度量值$M(s)$，因此$M(s)$是$T_{\text{sub}}(s)$上咬尾路径度量值的下界。

设路径P_{MLTB}的结束状态为s。对于集合$S_0$$(=S_L)$中的任意状态$s'$，其咬尾格形子图上的所有咬尾路径满足式(3)：$M_{\text{TB}}(s',s')\geq M(s')$。又$M_{\text{MLTB}}=\min(\{M(s),s\in S_L\})$，故由式(3)可得

$$M_{\text{TB}}(s',s')\geq M_{\text{MLTB}} \tag{4}$$

式(4)对S_0中任意状态s'皆成立，因此若P_{MLTB}是P中度量值最小的路径，则它是全局最优咬尾路径。

其次证明情况(2)和情况(3)。

针对情况(2)，P_{MLTB}不是P中度量值最小的路

径，集合 C 由所有 $M(s)$ 小于 M_{MLTB} 的状态组成。此时所有满足 $M(s) \geq M_{\mathrm{MLTB}}$ 的状态 s 都被从集合 C 中排除，这是因为在该状态的咬尾格形子图上的所有咬尾路径都满足式(3)，因而 $M_{\mathrm{TB}}(s,s) \geq M(s) \geq M_{\mathrm{MLTB}}$，这说明 $T_{\mathrm{sub}}(s)$ 上无法搜索到比 P_{MLTB} 度量值更小的咬尾路径，故状态 s 可以被删除。针对情况(3)，由于译码器初始化的时候 $M_{\mathrm{MLTB}}=+\infty$，所以集合 C 是 S_0 中所有状态的集合。对于情况(2)和情况(3)，需要用双向搜索在集合 C 中的咬尾格形子图上寻找可能存在的更优的咬尾路径。

如前所述，集合 C 中的状态是按照各个状态的 $M(s)$ 值按升序排列。VBS 算法在双向搜索过程中，先从 $M(s)$ 最小的状态 s 的 $T_{\mathrm{sub}}(s)$ 上开始搜索，通过不断调整门限值 Th 来搜索 $T_{\mathrm{sub}}(s)$ 上的咬尾路径。若门限值超过 M_{MLTB}，则说明状态 s 的 $T_{\mathrm{sub}}(s)$ 上没有比当前 P_{MLTB} 的度量值更小的咬尾路径，故可以停止在 $T_{\mathrm{sub}}(s)$ 上的搜索。如算法描述中所提，随着门限值增大，需要搜索的咬尾格形子图也越多。若 $\mathrm{Th} > M(s_{|C|})$，即门限值大于集合 C 中最后一个状态 $s_{|C|}$ 的 $M(s_{|C|})$ 值时，双向搜索需要遍历每个 $T_{\mathrm{sub}}(s)$，$\forall s \in C$。由于 $|S_0|$ 是一个有限的常数，而 $|C| \leq |S_0|$，因此双向搜索需要搜索的最大咬尾格形子图数是固定的，因此 VBS 算法是收敛的。由双向搜索算法的译码过程可知，它能够寻找到咬尾格形子图上小于当前门限值 Th 的最优的咬尾路径，并用来更新 $(P_{\mathrm{MLTB}},\ M_{\mathrm{MLTB}})$。当 $\mathrm{Th} \geq \mathrm{Th}_{\max} = M_{\mathrm{MLTB}}$ 时，译码结束，P_{MLTB} 中保存是全局最优解。

综上所述，VBS 算法是咬尾码在咬尾格形图上收敛的最大似然译码算法。　　证毕

3　性能仿真

本节通过两个实验来验证 VBS 算法的性能。并将 VBS 算法与 VH 算法[11]和基于双向搜索算法的双向有效码树搜索算法(Bidirectional Efficient Algorithm for Searching code Trees, BEAST)算法[10]进行比较。衡量算法的性能指标为译码过程中需要的存储空间大小和平均访问状态数。因为 3 种算法都是咬尾码的最大似然译码算法，误块率性能完全相同，故此处忽略误块率性能比较。3 种不同的算法对存储空间的需求计算方法分别为：

(1)VBS 算法存储空间需求：用于保存 Viterbi 算法获得全部幸存路径、咬尾格形图前向连接信息、咬尾格形图后向连接信息和双向搜索算法中断信息的存储空间；

(2)VH 算法存储空间需求：用于保存 Viterbi 算法获得全部幸存路径、全部幸存路径在每个时刻的度量值、咬尾格形图前向连接信息和启发式搜索中的开栈和闭合路径表的存储空间；

(3)BEAST 算法存储空间需求：用于保存咬尾格形图前向连接信息、咬尾格形图后向连接信息和双向搜索算法的中断信息的存储空间；

需要注意的是 BEAST 算法没有关于最大似然咬尾路径的任何信息，所以 BEAST 算法需要遍历所有 $T_{\mathrm{sub}}(s)\ (\forall s \in S_0)$，它需要访问的状态数要远远高于 VBS 算法访问的状态数。

实验 1　采用 LTE 中的咬尾卷积码$(90, 30)$。在译码过程中 3 种算法消耗的平均存储空间如图 3 所示，平均访问状态数如图 4 所示。从图 3 中可见，VBS 算法需要的存储空间是所有 3 种算法中最小的；BEAST 算法和 VBS 算法需要的存储空间相对稳定。随着信噪比的增大，VH 算法需要的存储空间会迅速下降，这是因为在启发式搜索算法中需要保存的开栈和闭合路径表大量减少。由图 4 可见，在译码过程中，VBS 算法和 VH 算法访问的状态数基本相同，而 BEAST 算法由于缺少每个咬尾格形图上的咬尾路径度量值信息，所以在遍历各个咬尾格形子图的过程中需要访问大量的状态数，复杂度非常高。

实验 2　采用码率为 1/2 的咬尾卷积码(48,24)，生成多项式为{113,147}。它是信息序列长度为 24 时的最小距离最大的咬尾卷积码生成多项式[14]。仿真中 3 种算法的平均消耗存储空间和平均访问状态数如图 5 和图 6 所示。我们可以得到和实验 1 中几乎相同的结论，同时可以发现，在信息序列较短时，BEAST 算法对存储空间的需求是少于 VBS 算法的，但是复杂度依然很高。

通过以上两个实验可以发现，VBS 算法相对于已有的最大似然译码算法在译码复杂度和存储空间消耗上有较好的折中，能以较低的复杂度和较少的存储空间得到相同译码结果。

4　结束语

在咬尾码的咬尾格形图上，由于起始状态的不确定性，Viterbi 搜索和传统双向搜索算法(如 BEAST 算法)需要做穷尽搜索，复杂度非常高；而 VH 算法虽然复杂度较低，但是消耗大量存储单元。本文结合 Viterbi 算法和双向搜索算法的优点，利用 Viterbi 搜索得到的最大似然咬尾路径度量值来排除不可能的起始状态，并将幸存路径度量值作为双向搜索的门限值上界。这样得到的新的最大似然译码算法，不仅译码复杂度较低，同时对存储空间的需求也很低，这对实际运用具有重要意义。

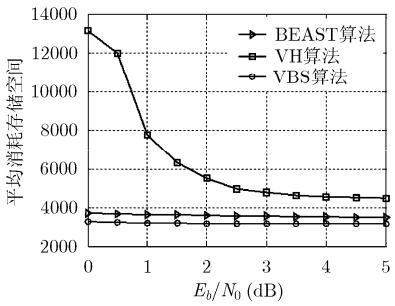

图 3 LTE 中咬尾卷积码译码需要的存储空间

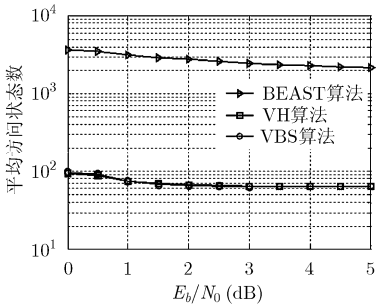

图 4 LTE 中咬尾卷积码
译码访问的平均状态数

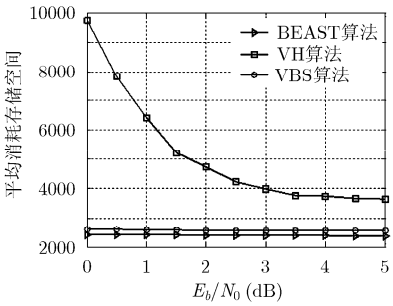

图 5 咬尾卷积码(48, 24)
译码时需要的存储空间

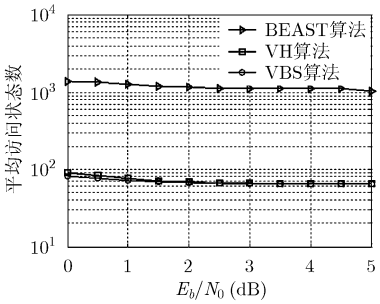

图 6 咬尾卷积码(48, 24)译码时需要的平均访问状态数

参 考 文 献

[1] Wang Xiao-tao, Qian Hua, Xu Jing, *et al.*. An efficient CVA-based decoding algorithm for tail-biting codes [C]. IEEE Globecom, Houston, 2011: 1–5.

[2] 王晓涛, 钱骅, 徐景, 等. 基于陷阱检测的咬尾卷积码译码算法[J]. 电子与信息学报, 2011, 33(10): 2300–2305.
Wang Xiao-tao, Qian Hua, Xu Jing, *et al.*. Trap detection based decoding algorithm for tail-biting convolutional codes [J]. *Journal of Electronics & Information Technology*, 2011, 33(10): 2300–2305.

[3] Wu T Y, Chen P N, and Pai H T, *et al.*. Reliability-based decoding for convolutional tail-biting codes[C]. IEEE Vehicular Technology Conference, Taipei, 2010: 1–4.

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

[5] 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.

[6] Calderbank A, Forney G Jr, and Vardy A. Minimal tail-biting trellises: the Golay code and more[J]. *IEEE Transactions on Information Theory*, 1999, 45(5): 1435–1455.

[7] Gluesing-Luerssen H and Weaver E. Linear tail-biting trellises: characteristic generators and the BCJR-construction[J]. *IEEE Transactions on Information Theory*, 2011, 57(2): 738–751.

[8] Gluesing-Luerssen H and Weaver E. Characteristic generators and dualization for tail-biting trellises[J]. *IEEE Transactions on Information Theory*, 2011, 57(11): 7418–7430.

[9] Solomon G and Tilborg H. A connection between block and convolutional codes[J]. *SIAM Journal on Applied Mathematics*, 1979, 37(2): 358–369.

[10] Bocharova I, Johannesson R, Kudryashov B, *et al.*. BEAST decoding for block codes[J]. *European Transactions on Telecommunications*, 2004, 15: 297–305.

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

[12] Pai H T, Han S, and Wu Y J. New HARQ scheme based on decoding of tail-biting convolutional codes in IEEE 802.16e [J]. *IEEE Transactions on Vehicular Technology*, 2011, 60(3): 912–918.

[13] 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.

[14] Stahl P, Anderson J, 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.

王晓涛： 男，1985 年生，博士生，研究方向为 LTE 物理层关键技术.

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.

康 凯： 男，1977 年生，博士后，高级工程师，研究领域包括无线通信系统物理层关键算法与实现、下一代无线网络等.