次最优软输出算法的高效存储策略

张 君,彭 华,石铠源
ZHANG Jun,PENG Hua,SHI Kai-yuan

信息工程大学 信息工程学院,郑州 450002
Information Engineering University,Institute of Information Engineering,Zhengzhou 450002,China
E-mail:zhangjun_smile@yahoo.com.cn

ZHANG Jun,PENG Hua,SHI Kai-yuan.Memory-efficient arrangement scheme for sub-optimal MAP algorithm.Computer Engineering and Applications,2009,45(2):116-118.

Abstract: The iterative receivers perform an outstanding reliable transmission at very low signal-to-noise ratio.However,the practical implementation of these is a very challenging topic,partly due to memory intensive.The approach of sliding window is used to reduce memory and delay.Applied the independent recursive nature of iterative decoding,this paper devotes an optimized arrangement scheme:the real-time release and update are proposed to reduce the memory consumption to 50%,the design of the computation for LLR without waiting is considered for reducing the delay to 33%,and there are still two backward recursion for remaining high performance.Simulation results support all above in the end.

Key words: iterative decoding;memory optimization;delay-reduced;high-efficient realization

摘 要:具有优秀误码性能的迭代接收机在实现中面临的一个关键问题就是译码算法对内存的过大需求。滑动窗法可以有效地减少内存以及处理时延。利用算法递归值的单次可利用性,进一步优化该法:通过对网格中状态度量进行实时的释放与更新,使内存消耗又缩减一倍;通过将似然比计算嵌入到双向递归的过程,使处理时延只为传统窗法的1/3;通过保留两个后向递归器结构,经实验表明接收机在优化实现后整体误码性能不减。

关键词:迭代译码;内存优化;减时延处理;高效实现
DOI:10.3778/j.issn.1002-8331.2009.02.033 **文章编号**:1002-8331(2009)02-0116-03 **文献标识码**:A **中图分类号**:TN914

1 引言

迭代信号处理技术已经成为现代通信领域的一个研究热点。接收机的优异性能来自于各个软输入软输出(SISO)模块间外信息的交换,尤其当所有接收比特(包括信息比特和校验比特)都参与迭代的情况。这使得基于比特最大后验概率(MAP)的软输出算法,如BCJR类算法[1],更适合迭代接收机。但是该类算法包括复杂的前向/后向递归过程,必须有大量的内存空间用于存储中间变量,尤其当编码器的状态数较多、输入符号帧较长时,若不加优化的直接用于硬件,几乎是不可实现的。所以从工程意义上讲,要解决迭代接收机的实用性问题,首要解决的就是低复杂度次优MAP算法的设计实现问题,这包括对运算量,存储空间和处理时延等因素的综合考虑。

然而在已有文献中,大多被关注的是算法运算量如何降低,对于高效存储和快速处理方面的讨论却较少。从硬件实现的角度看,后两者却是节省SISO模块能量消耗和反映算法实时性的关键环节[2]。近年来提出的一些改进策略,主要是从算法本身的减状态过程推导[3]和改变存储单元的分配策略[2,4-6]两方面进行优化。后者大致也可归为两类。一是根据状态度量值被调用的特点,令在逐步得到前向递归值(或后向递归值)的同时,调出内存中反向递归的量值,而自身不加存储的直接用于计算当前时刻似然比并输出,如后向度量等待的WAB(Write After Block read)策略,其时延保持不变,而内存利用率可提高一倍;另一类则借鉴滑动窗(SW)机制,以化整为零的思想使得存储长度仅与等价VA算法中共生路径的发生长度有关[4-5],输出时延也远远小于整帧长的处理时间。最近,文献[6]在第一类思想的基础上,给出了一种双向递归同时进行的高效信息更新方法,使得译码速度又获得了约50%的提高。

结合上述思想,针对Max-Log-MAP算法[1]设计了一种新的高效存储并快速处理的改进策略。该策略是对已有内存方案的进一步优化,体现了优势综合;且具有一般性,对基于网格结构的以前向/后向递归实现的算法均可适用。

2 MAP算法描述

为简便,下文以符号 X_L^1 表示序列$\{x_1,x_2,\cdots,x_L\}$。

记 $U_1^{L_b}$ 为编码后的二值比特帧(L_b 为比特帧长),$Y_1^{L_b}$ 为受噪声污染的接收实值符号帧。对应的编码网格共有 K 个网格时刻($K=L_b\cdot R$,R 为单输入编码的码率;网格时刻定义为一次状

作者简介:张君(1983-),女,硕士,主要研究方向为解调译码联合处理技术、信道编译码技术等;彭华(1973-),男,博士,硕导,副教授,主要研究领域为软件无线电关键技术等;石铠源(1981-),男,硕士,主要研究领域为通信中的信号处理技术。

收稿日期:2007-12-28 **修回日期**:2008-03-17

态转移的到达时刻),第 k 个时刻的第 i 个比特其后验概率可以对数似然比的形式表述为

$$L_i \approx \max_{U_1^{L_b}:u_i=+1} M(\boldsymbol{U}_1^{L_b},\boldsymbol{Y}_1^{L_b}) - \max_{U_1^{L_b}:u_i=-1} M(\boldsymbol{U}_1^{L_b},\boldsymbol{Y}_1^{L_b}) = M_{+1}[i]-M_{-1}[i]$$
$$(k=1,2,\cdots,K;i=2k-1,2k) \quad (1)$$

式中利用了 Jacobian Logarithm 的近似公式得到简化的算法,其中 $M(U_1^{L_b},Y_1^{L_b})$ 表示软输出的路径量度值,定义为

$$M(U_1^{L_b},Y_1^{L_b}) \triangleq Inp(U_1^{L_b},Y_1^{L_b}) = Inp(Y_1^{L_b}|U_1^{L_b}) + Inp(U_1^{L_b}) \quad (2)$$

由式(1)(2)看出,计算每一比特的似然比需要 2^{L_b} 个乘积项相加,运算量很大;不过根据码字的网格结构,可以通过转移分支的前向递归和后向递归来简单运算,其中前向度量表示为

$$\alpha_k^*(s)=\max_{s'\in S_k}[\gamma_k^*(s',s)+\alpha_{k-1}^*(s')] \quad k=1,2,\cdots,K \quad (3)$$

后向度量表示为

$$\beta_k^*(s')=\max_{s\in S_{k+1}}[\gamma_k^*(s',s)+\beta_{k-1}^*(s)] \quad k=K-1,K-2,\cdots,0 \quad (4)$$

式中 $\gamma_k(s',s)=p((y_{2k-1},y_{2k}),s|s')$ 为分支度量,与当前输入的符号对有关;星号标记出概率的对数域表示;s' 和 s 分别对应前一时刻和当前时刻网格中所处的状态,S_k 表示在时刻 k 所有可能状态的集合。

则式(2)可表示为

$$M(U_1^{L_b},Y_1^{L_b})=\alpha_{k-1}^*(s')+\gamma_k^*(s',s)+\beta_k^*(s) \quad (k=1,2,\cdots,K) \quad (5)$$

综上,结合串行迭代的接收机结构,计算第 $2k-1$(或 $2k$)个编码比特的似然比包括下述 4 个步骤:

步骤 1　分别从前向和后向递归存储中取出所有状态在时刻 k 的前向度量值和时刻 $k+1$ 的后向度量值;

步骤 2　依次从每一状态出发,由有限状态机(FSM)选择发送信息为 1 或 0 时在时刻 $k+1$ 可能到达的状态,计算各转移分支的度量;

步骤 3　由式(5)计算使得编码比特 u_k 为+1 时所有可能路径的度量,比较并选出最大值,记为 $M_{+1}[i]$;同理计算出 $M_{-1}[i]$;

步骤 4　由式(1)得到 L_i。

3　算法实现策略

3.1　基础策略

由上述算法流程可以看出,最直接的实现需要两个等帧长的内存空间,分别存储所有状态在每一时刻的前向度量值和后向度量值。若用 S 表示网格的状态数,定义存储一个状态度量值所需的寄存器组为一个内存单位,则该内存空间的量值为

$$Mem_{Orig}=2*K*S \quad (6)$$

相应的,若定义算法在一次单向处理中 K 次状态转移的平均运算耗时为一个时单位,则输出总时延包括一个前向递归过程(同时进行的后向递归过程)和一个前向的似然比计算过程:约有 $2K$ 个时单位。图 1 为该策略的简单示意。

3.2　滑动窗策略

本文的改进策略是在文献[4-5]的基础上完成的。文献[4]中对后向递归的处理过程是滑动窗法在内存分配中的一种典型应用,它需要一个等帧长的前向度量存储空间(下文简称前向内存)和一个滑动窗控制的后向度量存储空间(下文简称后向内存);通过增加一个后向递归器保持了算法的误码性能(参见图 2)。文献[5]则是该应用结合硬件实现并进一步减少内存

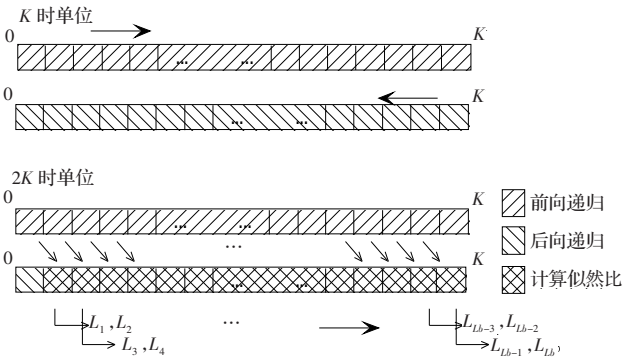

图 1　基础实现策略示意图

消耗的优化方案,它保留了文献[4]中所有的优势策略,并利用了窗输出结果能为下一窗前向递归提供出可靠的起始状态的特点,将前向度量也存储在滑动窗中(由于仅此差别,文中不再给出相应的示意图);通过窗的重复使用完成了对整帧数据的存储与处理;整个实现的内存消耗量仅相当于文献[4]中两个滑动窗的大小。

下面结合文献[5]中的优化策略具体说明。首先描述其理论基础。该策略将 Max-Log-MAP 算法中前向和后向递归的过程看作两个独立的 VA 过程,并应用了 VA 算法的"冷启动"特性,即路径存储时间 τ 大于 6 倍编码约束长度 L 时,所有 S 条路径总会以接近于 1 的概率源自同一公共干路[7]。定义 $T=\min\{\tau\}=6L$,称为共生路径的发生长度。这样就可以认为,在起始状态不确定的情况下经 $\tau(\tau>T)$ 步递归得到的后验度量值与起始自时刻 T 的某一确定状态又经 $\tau-T$ 步递归得到的度量值可靠度相当。

于是,为了得到较准确的后向度量,算法引入了两个后向递推器,以相错 T 的起始位置同时运算。在最初的 T 时段,各递归器仅为了得到 T 时刻的"确定"状态,故计算结果均不记入内存。从 $T+1$ 时刻开始,才分别将"可靠"的后向度量存入滑动窗。不难得到滑动窗长为 $2T$。

对于前向递归器,它于相同的起始时刻由确定的全零状态独立运算,并将计算结果记入另一滑动窗空间;$2T$ 长度的数据处理完毕,与后向内存中的相应度量值共同计算各个时刻的似然比;完毕即释放内存,等待下一 $2T$ 长度的数据输入。

由上看出,实现中需要两块长度为 $2T$ 的内存空间,分别记录各个时刻全部可能状态的前向度量值和后向度量值。类似于式(6),内存空间的量值可表示为

$$Mem_{SW}=2*2T*S \quad (7)$$

式中,由于 $T<<K$,内存空间必然大大减小。

考察上述过程不难发现,总时延与窗时延成正比。分析每一滑动窗的处理过程,它包括一个 $2T$ 长度的前向递归过程,一个等价的 $2T$ 长度的后向递归过程和一个似然比计算过程。故输出时延为 $6T$ 个时单位,一帧的总处理时延为 $3K$ 个时单位。

总结该方案,它通过增加一个后向计算单元,获得了加窗后内存空间减为原有的 $2T/K$,并保持可靠度不降低的良好性能。尤其 $T<<K$ 时,节省的内存空间相当可观。但应注意到,该法却引入了额外的时延。

图 2 中,空白部分表示空间未被利用,虚线框部分表示两个后向递归器的产生结果只用于后续计算而不记入内存。

3.3　本文的改进策略

事实上,MAP 算法具有单次利用性,即状态度量(包括前

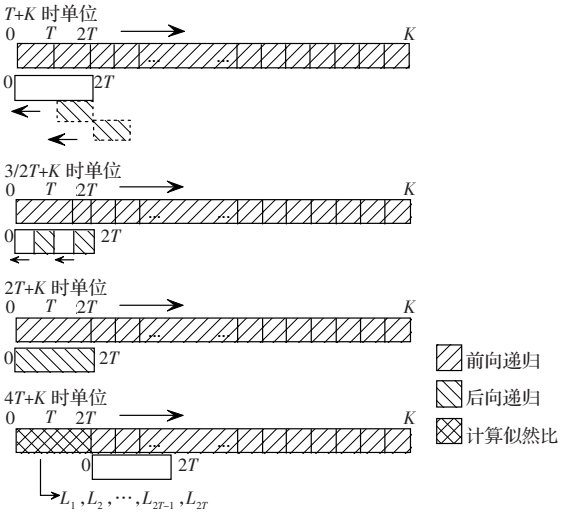

图 2　传统的滑动窗策略

向和后向)一经利用就可被马上释放,而不影响后续进程。所以考虑在传统的滑动窗策略中设计一种方案使得前向/后向度量在窗内能够共享内存，这样就可进一步获得 $S*2T$ 空间的大幅缩减。同时,为了达到最小输出时延的目的,文中将双向信息更新的方法引入到新策略中,实现在 $2T$ 时单位起开始输出,仅需 K 个时单位的总处理时延就能够获得整帧的似然比值。

下面以一个接收符号帧的第 r 段滑动窗数据为例具体说明。

首先初始化。分得内存空间 $S*2T$。对于接收符号帧,三个递归器分别定位于各自的运算起始时刻。其中,前向递归器对应于确定的 $r\cdot T$ 时刻,第一个后向递归器对应 $(r+3)T$ 时刻,第二个后向递归器对应 $(r+2)T$ 时刻。应当说明,对于后向递归器其起起始时刻所有状态被视为等概发生。

各递归器同时启动。根据对应于时单位 p 的不同操作,将计算过程分为 3 个阶段。

第一阶段:$p\leqslant T$

前向递归器计算 α_p 并存入第 p 个内存列单元;两个后向递归器只计算递推结果但不存入内存;

第二阶段:$T+1\leqslant p\leqslant 3/2T$

前向递归器仍将计算出的 α_p 存入第 p 个内存列；第一个后向递归器将此时的计算结果作为 β_{3T-p+1} 存入空间第 $3T-p+1$ 个内存列;第二后向递归器则在计算出 β_{2T-p+2} 的同时,与内存中已有的 α_{2T-p+1} 利用式(5)共同计算第 $2T-p+1$ 时单元的各路径度量,最终计算得到似然值 L_{2k-1},L_{2k},并存入相应时刻 k 的内存列中(注意到,总有 $S\geqslant 2$);

第三阶段:$3/2T+1\leqslant p\leqslant 2T$

类似上述,前向递归器和两个后向递归器在此阶段均需在计算出递归度量同时,利用内存中的已有值计算最终输出的编码比特的似然比值;至 $2T$ 时刻,该窗内的数据运算结束,结果值依次输出。

图 3 为此过程的示意图。可以得到该设计方案仅需 $S*2T$ 的内存空间,经 $2T$ 个时单位就可完成窗内的计算,经 $2K$ 个时单位就可输出整帧数据对应的似然比值。

4　系统仿真与结果分析

为了验证文中改进策略的误码性能,建立迭代接收机的系统仿真平台。其中,信源采用先验统计特性不知的 0/1 二值随机序列;伪随机交织器;信道编码采用(2,1,2)系统递归卷积

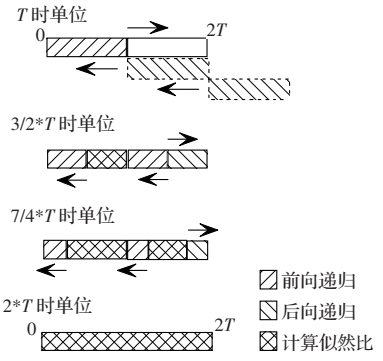

图 3　内存分配策略示意图

码,生成多项式$(7\ \ 5)_8$,码率 1/2;信号分别采用 8PSK 调制和 16QAM 调制方式;信道采用无码间干扰的 AWGN 信道。

图 4 和图 5 表明接收机分别采用本文改进的软译码实现策略和未经处理的基础策略而得到的系统误比特性能,并表示为 E_b/N_0 的函数(E_b 为信息比特的平均能量,N_0 为单边功率谱密度)。正如前文分析,两个后向递归器的巧妙设计避免了滑动窗带来的性能损失。由图看出,即使结合高阶调制,系统的整体性能也几乎没有改变。

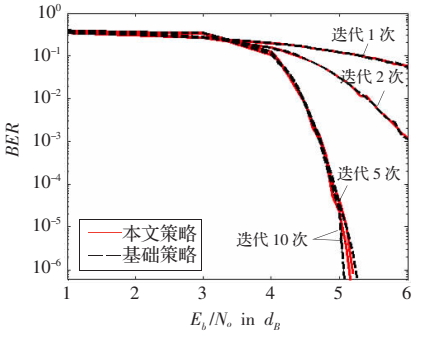

图 4　迭代接收机误码性能曲线(8PSK)

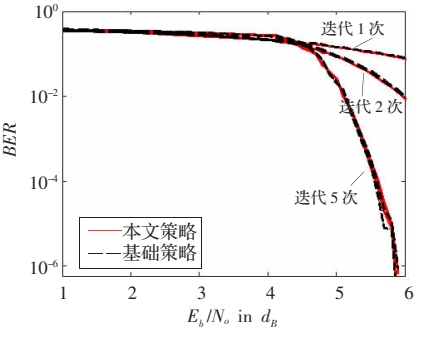

图 5　迭代接收机误码性能曲线(16QAM)

进而,通过表 1 对前述三种策略在内存和时延量值上的总结与比较,更便于表明本文方案的改进效果。虽然与传统滑动窗法相比,文中策略没有量级上的大差别,但当资源约束较为苛刻、待处理的数据量很大时,改进优势得以很好体现。

表 1　三种方案内存和时延的量化比较

	存储空间(内存单元)	输出时延(时单位)	帧处理时延(时单位)
基础策略	S*2K	2K	2K
传统窗策略	S*2T	6T	3K
本文策略	S*T	2T	K

(下转 136 页)