LT码的BPML译码算法

朱宏鹏 李广侠 冯少栋

(解放军理工大学通信工程学院 南京 210007)

摘 要 采用置信度传播算法(BP)对LT码进行译码时,停止集是影响译码效率的重要因素。对LT码停止集的大小进行了理论分析和仿真,提出了置信度传播-最大似然联合译码算法(BPML)。该算法首先采用BP算法译码,当遇到停止集时再采用最大似然译码算法(ML)对停止集进行处理,能够有效消除停止集的影响,提高LT码的译码效率。仿真结果表明,BPML算法结合了BP算法复杂度低和ML算法译码效率高的优点。研究结果对提高计算机网络中数据分发应用的分发效率具有重要的实用价值。

关键词 数据分发,喷泉码,LT码,BPML,停止集

中图法分类号 TN911.22 **文献标识码** A

BPML Decoding Algorithm of LT Codes

ZHU Hong-peng LI Guang-xia FENG Shao-dong

(Institute of Communication Engineering, PLA University of Science and Technology, Nanjing 210007, China)

Abstract For Belief Propagation(BP) decoding algorithm of LT codes, stopping set prohibits the improvement of decoding efficiency. This paper analyzed and simulated the size of stopping set. A Belief Propagation-Maximum Likelihood decoding algorithm(BPML) was proposed. BPML uses BP algorithm to decode firstly. When stopping set makes BP stop, Maximum Likelihood(ML) decoding algorithm is used to deal with the stopping set. It can overcome the negative influence of stopping set and improve the decoding efficiency of LT codes. The simulation showed that BPML combines the advantages of BP algorithm in low decoding complexity and ML algorithm in high decoding efficiency. The conclusion of research is practically valuable in improving efficiencies of data distribution applications in computer networks.

Keywords Data distribution, Digital fountain, LT codes, BPML, Stopping set

1 引言

Elias在文献[1]中提出二进制删除信道(BEC)模型,证明了删除概率为 p 的BEC信道容量为 $1-p$。计算机网络中,由于节点拥塞、链路失效等原因,会出现分组丢失,其信道是典型的BEC。Elias证明了采用分组级前向纠错编码(FEC)技术并进行最大似然(ML)译码,能够使通信速率趋近于信道容量。

传统的分组级FEC技术,如RSE码,是一种固定码率编码。编码之前,编码器需要知道信道的分组丢失特性,以设置合适的码率。在组播和广播等数据分发应用中,发送端同时与多个接收端进行通信,从发送端到每个接收端的信道特性不尽相同。为了保证所有接收端都能够正确接收,发送端需要根据最差的信道来设置码率,这就造成了信道条件较好的接收端接收效率的下降。译码时采用ML算法,等效于求解线性方程组。用高斯消元法求解方程组,复杂度随码长呈平方关系增长。当码长较大时,译码时间很长。

喷泉码是近几年提出的一类新的分组级FEC技术[2],主要包括Tornado码[3]、LT码[4]和Raptor码[5]等。该技术采用随机编码思想,编码码率动态可变,在有限数目的原始数据分组输入的情况下可以产生无限数目的编码分组。接收端在收到任意一组稍多于原始数据分组总数的编码分组后,就能正确恢复出所有的原始数据分组,而不管具体接收到的是哪些编码分组。喷泉码能够保证任意数量信道特性异构的用户在任意时刻接入系统,都能以接近于点对点通信的效率完成数据的接收[2],适用于组播和广播应用。喷泉码采用置信度传播(BP)算法进行译码,与ML算法相比,降低了译码复杂度,缩短了译码时延。目前,喷泉码已经被第三代蜂窝网络多媒体广播/多点传送服务(MBMS)和DVB-H标准(手机电视标准)所采用[6]。

LT码是第一次真正意义上实现数字喷泉思想的编译码技术。M. Luby提出采用BP算法对LT码进行译码[4]。当遇到停止集时,BP算法停止译码,不能充分挖掘编码分组关于停止集中原始数据分组的信息。因此,停止集的存在是影响BP算法译码效率的重要因素。本文对LT码的停止集进行研究,提出一种能够克服停止集负面效应的置信度传播-最大似然联合译码算法(BPML),以提高LT码的译码效率。

到稿日期:2008-11-11 返修日期:2009-07-23

朱宏鹏(1982—),男,博士生,主要研究方向为数字喷泉、网络编码、组播和广播通信等,E-mail:zhl_dgqb@126.com;李广侠(1964—),男,博士生导师,主要研究方向为卫星通信、网络通信;冯少栋(1981—),男,博士生,主要研究方向为宽带多媒体通信。

2 LT 编译码原理

在此对文中所用的参数作如下说明：原始数据分组的数目为 k，接收端参加译码的一共有 P 个编码分组，编码分组的度为 d。

2.1 LT 编码[4]

LT 码编码过程如下：

1)欲产生一个编码分组时，首先按照度分布函数随机选择一个度 d。

2)从所有的原始数据分组中等概随机地选取 d 个分组作为生成该编码分组的原始数据分组。

3)将选取的 d 个原始数据分组进行异或运算，得出的结果便是编码分组的值。

重复以上步骤可以源源不断产生编码分组。

度分布函数选择次优度分布[7]，该分布与稳健孤立子度分布[4]相比具有更高的译码效率和更加稳定的性能。

定义 1(度分布函数) 对于所有的度 d，度分布函数 $\rho(d)$ 是编码分组度为 d 的概率。

定义 2(次优度分布)

$$\rho(1)=(R+1)/k$$

$$\rho(i)=\frac{k}{P}\cdot\frac{1}{i(i-1)}\cdot\sum_{L=R+1}^{k-1}\frac{L}{L-R}/(k-R-1),\quad i=2,\cdots,k-R \tag{1}$$

$$\rho(k-R+1)=C\cdot R\ln(R/\delta)/((k-R+1)\cdot P)$$

其中，$R=2+8\cdot\sqrt[4]{k}$[8]，C 为常数因子，δ 为允许的译码失败概率。

上面第二步在随机选取 d 个原始数据分组时，如果完全随机选择，为了在编码分组的报头信息中传送原始数据分组的位置信息，需要占用 $d\cdot\log 2k$ 比特的开销，其中 k 为原始数据分组的数目。当 k 和 d 较大时，此项开销很大，在实际应用中并不可取。因此文献[9]给出了实用的有限随机 LT 码的构造方法，该方法可大大节省报头开销。

2.2 LT 译码

LT 译码对接收到的 P 个编码分组进行处理，恢复出 k 个原始数据分组。评判译码算法的主要标准是译码效率和译码复杂度。译码复杂度通常由译码时间来表征。

定义 3(译码效率) 译码器恢复 k 个原始数据分组需要 P 个编码数据分组，令 e 表示译码效率，则

$$e=k/P \tag{2}$$

LT 码的译码算法主要包括 BP 算法和 ML 算法。

2.2.1 BP 译码算法

定义 4(BP 译码算法[4]) 开始时 k 个原始数据分组均未恢复。首先，释放 P 个编码分组中所有度数为 1 的分组，恢复出它们所对应的原始数据分组。已经被恢复出来但还未被处理的原始数据分组的集合构成预处理集。译码过程中每一步处理操作如下：从预处理集中选出一个原始数据分组，将该分组与 P 个编码分组中尚未被释放的且和它相关的分组进行异或运算。所有参加异或运算的编码分组度数减 1，异或之后度数变为 1 的编码分组被释放，所对应的原始数据分组被恢复出来，并加到预处理集当中。刚刚处理的原始数据分组从预处理集当中剔出，而处理之后新恢复的原始数据分组如果不在预处理集当中，就会引起预处理集的增长，反之则不会引起预处理集的增长。当预处理集为空时，即没有可以被处理的原始数据分组时，此过程结束。译码失败是指在所有的原始数据分组恢复出来之前预处理集就已经变空。

为了便于理解，下面通过图 1 的例子对 BP 译码过程进行描述。

假设对 3 个原始数据分组进行 LT 编码后产生 4 个编码分组，如图 1(a)所示，其中 S_1，S_2 和 S_3 为原始数据分组，E_1，E_2，E_3 和 E_4 为编码分组。原始数据分组与编码分组之间的连线表示异或运算，如 E_2 是由 S_1，S_2 和 S_3 异或产生。开始译码时，首先寻找所有度数为 1 的编码分组，即 E_1，恢复出它对应的原始数据分组 S_1，并将 S_1 加入到预处理集当中，如图 1(b)所示。接下来将 S_1 与 E_2 和 E_4 进行异或运算，E_2 和 E_4 的度都减 1，变为 2 和 1，然后将 S_1 从预处理集当中剔出，如图 1(c)所示。从尚未释放的编码分组 E_2，E_3 和 E_4 中找出度数变为 1 的 E_4，恢复出其所对应的 S_2，并将 S_2 加入到预处理集中，如图 1(d)所示。将 S_2 与 E_2 和 E_3 进行异或运算，E_2 和 E_3 的度均变为 1，然后将 S_2 从预处理集当中剔出，如图 1(e)所示。紧接着恢复出 E_2 和 E_3 对应的原始数据分组 S_3，如图 1(f)所示。所有的原始数据分组均被恢复，译码过程结束。

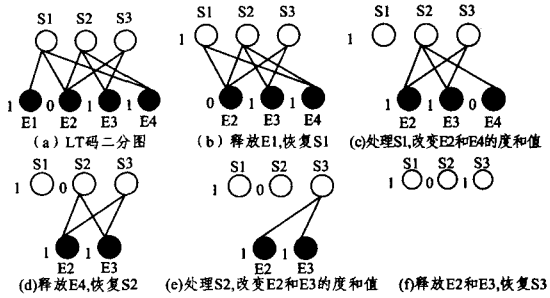

图 1 LT 码的 BP 译码过程

停止集能够导致 BP 算法译码失败。下面介绍停止集的概念，举例说明其对译码的影响，并从理论上分析其大小。

定义 5(停止集) 如果一组原始数据分组相关的编码分组的度都大于 1，则这组原始数据分组构成的集合称为停止集。

以图 1(a)为例，如果编码分组 E_1 丢失，则对应的二分图如图 1(b)所示，其中 S_1 的值未知。此时 S_1，S_2，S_3 构成一个停止集，因为其相关的 E_2，E_3，E_4 的度均大于 1。采用 BP 译码算法，当 E_1 丢失时，由于找不到度数为 1 的编码分组，无法继续译码，导致译码失败。

假设 LT 码的二分图包含 k 个原始数据分组，P 个编码分组，编码分组度数为 d 的概率 $\rho(d)$，其相邻的 d 个分组从 k 个原始数据分组中随机选取，以此种方法构成的二分图的集合用 $G(\rho,k,P)$ 表示。对于 $G(\rho,k,P)$ 中的二分图，下面的定理给出最大停止集的大小为 s 的近似概率。

定理 1 令

$$A_0(P,0)=1$$

$$A_0(z,o)=0,(z,o)\neq(P,0) \tag{3}$$

$$A_{n+1}(z,o)=\sum_{l,m}A_n(l,m)\cdot\sum_d\Lambda(d)\cdot f,n\geqslant 0$$

其中，$f=\binom{l}{l-z}\binom{m}{m+l-z-o}\binom{P-l-m}{d-m-2l+2z+o}/\binom{P}{d}$。

$z\geqslant 0,o\geqslant 0,d\geqslant 1$，$P$ 是编码分组的数目，$\Lambda(d)$ 是编码分

组度数为 d 的概率。

$A_n(z,o)$表示 $G(\rho,k,P)$中的二分图有 z 个度数为 0 和 o 个度数为 1 的编码分组的概率，则 $G(\rho,k,P)$中的二分图最大停止集的大小为 s 的概率约为

$$\Pr ob_{\text{stop}}(s)\approx\binom{k}{s}\sum_{z=0}^{r}A_s(z,0)\left(1-\sum_d\Lambda(d)\binom{r-z}{d}/\binom{r}{d}\right)^{n-s} \tag{4}$$

定理 1 的证明类似于文献[5]中定理 6 的证明，在此仅说明如何由编码分组度的分布推出原始数据分组度的分布，即由 $\rho(i)$如何推出 $\Lambda(d)$。编码分组度数为 i 的概率为 $\rho(i)$，a 是编码分组的平均度数，$a=\sum i\rho(i)$。每个编码分组从 k 个原始数据分组中等概随机选取相邻分组，因此原始数据分组有 d 个相邻编码分组的概率为 $\binom{P}{d}(a/k)^d(1-a/k)^{P-d}$。

令停止集的势的均值为 $\overline{N}_{\text{StopSet}}$，则

$$\overline{N}_{\text{StopSet}}=\sum_{s=1}^{k}s\cdot\Pr ob_{\text{stop}}(s) \tag{5}$$

从以上定理可以看出，LT 码中停止集的理论分析十分复杂。

BP 算法的译码复杂度与 LT 码二分图中边的数目成正比，运算量为 $O(k\cdot\ln k)$[4]。

2.2.2 ML 译码算法

LT 码是线性码，采用 ML 算法译码的本质是求解线性方程组，其优点是能够利用所有编码分组的信息，从而获得最高的译码效率。

定义 6(ML 译码算法[10]) 假设接收端接收到 P 个编码分组，首先构造一个 $P\times k$ 的矩阵 M，所有的元素都初始化为 0。接下来依次对 P 个编码分组进行处理。在处理第 i 个编码分组时，假设该编码分组的度为 d，其相关的原始数据分组分别为 $S_{i1},S_{i2},\cdots,S_{id}$，则将矩阵 M 中第 i 行的第 $i_1,i_2,\cdots,i_d$ 个元素置为 1，该行的其它元素仍为 0。依此类推，直到处理完 P 个编码分组。通过求解下式，可以恢复出 k 个原始数据分组：

$$M\cdot\vec{S}'=\vec{E}' \tag{6}$$

其中，$\vec{S}$是 k 个原始数据分组的值所组成的行向量，$\vec{S}'$表示$\vec{S}$的转置，$\vec{S}=S_1,S_2,\cdots,S_k\}$，$\vec{E}$是 P 个编码分组的值所组成的行向量，$\vec{E}=\{E_1,E_2,\cdots,E_P\}$。

通过高斯消元法求解式(6)，只要矩阵 M 的秩为 k，就可以完全恢复出 k 个原始数据分组。

以图 1(a)所示的二分图为例，$\vec{E}=\{1\quad 0\quad 1\quad 1\}$，则式(6)可表示为

$$\begin{bmatrix}1&0&0\\1&1&1\\0&1&1\\1&1&0\end{bmatrix}\cdot\begin{bmatrix}S_1\\S_2\\S_3\end{bmatrix}=\begin{bmatrix}1\\0\\1\\1\end{bmatrix} \tag{7}$$

由上式可以求出$\vec{S}$，$\vec{S}=\{1\quad 0\quad 1\}$。

该算法的缺点是译码复杂度高，运算量为 $O(Pk^2)$，远大于 BP 算法的运算量。

3 BPML 译码算法

BP 算法具有译码复杂度低的优点，但停止集会导致译码过程不能充分利用编码分组的信息，降低了译码效率。ML 算法不受停止集的影响，能够挖掘编码分组的所有信息，与 BP 算法相比提高了译码效率，但译码复杂度较高。本节提出 BPML 算法，该算法能够结合 BP 算法复杂度低和 ML 算法译码效率高的优点。

定义 7(BPML 译码算法) 首先采用 BP 算法对 P 个编码分组进行译码，如果能够恢复出所有的原始数据分组，则译码结束。倘若出现停止集，导致 BP 算法无法继续译码时，将尚未恢复的原始数据分组和尚未释放的编码分组组成 LT 码约简二分图，再对该约简二分图进行 ML 译码。ML 译码结束后，BPML 译码完成。

下面通过图 2 所示的例子来阐述 BPML 译码过程。

假设对 5 个原始数据分组进行 LT 编码产生 6 个编码分组，如图 2(a)所示。开始译码时，首先寻找所有度数为 1 的编码分组，即 E_1，恢复出它对应的原始数据分组 S_1，将 S_1 加入到预处理集当中，如图 2(b)所示。接下来将 S_1 与 E_2 和 E_4 进行异或运算，E_2 和 E_4 的度都减 1，变为 1 和 2，然后将 S_1 从预处理集当中剔出，如图 2(c)所示。从尚未释放的编码分组 E_2 和 E_4 中找出度数变为 1 的 E_2，恢复出其所对应的 S_2，并将 S_2 加入到预处理集中，如图 2(d)所示。将 S_2 与 E_3 和 E_6 进行异或，E_3 和 E_6 的度均变为 2，然后将 S_2 从预处理集当中剔出，如图 2(e)所示。由 S_3，S_4 和 S_5 所引出的二分图是一个停止集，如图 2(f)所示。

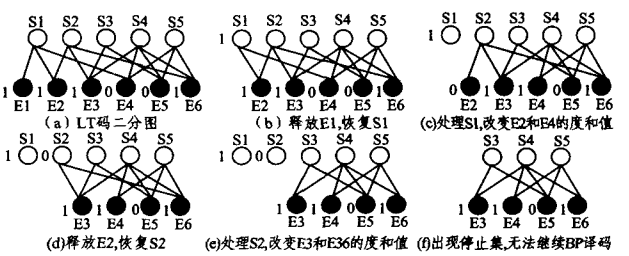

图 2 LT 码的 BPML 译码过程

BP 算法若找不到度数为 1 的编码分组，则无法继续译码。此时，采用 ML 算法对停止集进行处理。

ML 算法首先构造系数矩阵 M，由图 2(f)的连接关系可以得出 M，如式(8)所示：

$$M=\begin{bmatrix}1&1&0\\0&1&1\\1&1&1\\0&1&1\end{bmatrix} \tag{8}$$

矩阵 M 的秩为 3，由式(9)可以求出 S_3，S_4 和 S_5 的值。

$$\{S_3\quad S_4\quad S_5\}\cdot M'=\{E_3\quad E_4\quad E_5\quad E_6\} \tag{9}$$

与 BP 算法相比，BPML 算法消除了停止集对译码的影响，利用了编码分组中包含的关于停止集内原始数据分组的信息，提高了译码效率。与 ML 算法相比，BPML 算法采用 BP 算法恢复出大多数原始数据分组，只对停止集内较少的原始数据分组采用 ML 算法，缩小了 ML 译码矩阵的规模，减小了运算复杂度。

4 性能仿真与分析

本文首先对停止集的大小进行仿真，然后对 BP 算法、ML 算法以及 BPML 算法的译码效率和译码时间进行仿真比

较。

4.1 停止集仿真

停止集是影响 BP 算法译码效率的主要因素，其理论分析过于复杂，本节通过计算机仿真来评估它的大小。

首先给出译码开销的定义。

定义 8(译码开销) 接收端用 P 个编码分组来恢复 k 个原始数据分组，令 $1+\varepsilon$ 表示译码开销，则

$$1+\varepsilon=P/k \tag{10}$$

仿真中，原始数据分组数目 k 分两段：第一段在 100～900 之间，以 100 为步进值，第二段在 1000～10000 之间，以 1000 为步进值。译码开销 $1+\varepsilon$ 在 1.0～1.4 之间，以 0.05 为步进值。对于 k 个原始数据分组，选取 $(1+\varepsilon)/k$ 个编码分组，采用 BP 算法进行译码。当 BP 译码停止时，查看尚未恢复的原始数据分组的数目，即停止集的大小 N_{stopset}。停止集的归一化大小 N_{stopset}/k 表示 BP 算法尚未恢复的原始数据分组的比例。每一个参数点仿真 200 次，统计 200 次仿真结果的均值。

原始数据分组数目 k 在 100～900 之间的仿真结果如表 1 所列，停止集的归一化大小如图 3 所示。k 在 1000～10000 之间的仿真结果如表 2 所列，停止集的归一化大小如图 4 所示。从仿真图表可以看出，当译码开销为 1 时，有一半左右的原始数据分组无法通过 BP 算法恢复。随着译码开销的增加，停止集迅速减小。k 在 100～900 范围内，停止集减小相对较缓，译码开销为 1.4 时，所有的曲线才基本趋近于 0 点；k 在 1000～10000 之间时，停止集减小较快，译码开销为 1.15 时，所有的曲线基本都已趋于 0 点。增加译码开销可以减小停止集，提高 BP 算法译码成功的概率，代价是降低了译码效率。从图中还可以看出，当 $1+\varepsilon>1.05$ 时，同样的译码开销，k 越大，归一化停止集就越小，BP 算法的译码性能也就越好。LT 码基于随机编码思想，k 的增加使其随机性更好，同样的译码开销能够包含更多比例的原始数据分组的信息，所以随着 k 的增加，更多比例的原始数据分组能够被恢复出来。

停止集的存在是提高 BP 算法译码效率的瓶颈。如 k 为 10000 时，要使停止集的均值小于 1，仍需 1.1 的译码开销，译码效率只能达到 0.9091。因此，消除停止集的影响，对提高 LT 码的译码效率至关重要。

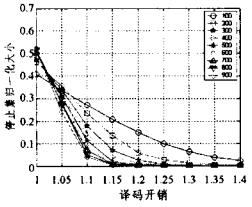

图 3 停止集归一化大小(k：100～900)

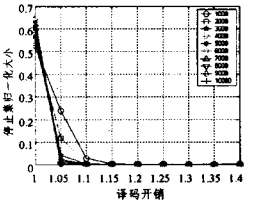

图 4 停止集归一化大小(k:1000～10000)

表 1 停止集大小统计(k:100～900)

译码开销	1.0	1.05	1.1	1.15	1.2	1.25	1.3	1.35	1.4
译码效率	1.00	0.9524	0.9091	0.8696	0.8333	0.8000	0.7692	0.7407	0.7143
100	40.735	33.583	27.173	20.69	15.139	9.9414	6.4406	3.7516	2.392
200	92.498	70.921	47.047	27.058	11.841	5.5228	2.5194	1.1936	0.4906
300	141.72	101.5	53.77	20.881	7.5388	2.4032	1.1522	0.424	0.2474
400	190.72	126.31	57.635	16.076	5.1384	1.4396	0.7912	0.4396	0.1786
500	249.72	153.89	56.759	10.22	2.8088	1.3898	0.6638	0.162	0.047
600	299.74	178.42	50.624	7.5956	2.4396	0.5056	0.2234	0.0642	0.056
700	354.71	195.91	49.316	7.12	1.4174	0.8332	0.1744	0.0164	0.0164
800	413.38	222	40.903	6.2842	1.219	0.3808	0.1776	0.0282	0
900	466.8	216.76	35.911	5.441	1.6494	0.4002	0.24	0.1144	0

表 2 停止集大小统计(k:1000～10000)

译码开销	1.00	1.05	1.10	1.15	1.20	1.25	1.30	1.35	1.40
译码效率	1.00	0.9524	0.9091	0.8696	0.8333	0.8000	0.7692	0.7407	0.7143
1000	527.41	239.26	28.165	3.25	1.38	0.375	0.255	0.16	0.04
2000	1074.9	234.81	7.16	1.22	0.365	0.32	0.14	0	0
3000	1695.9	123.39	3.5	1.055	0.345	0.045	0.045	0	0
4000	2387.5	87.545	3.555	0.685	0.36	0.195	0.2	0	0
5000	3142.2	103.82	2.55	0.39	0.1	0	0	0	0
6000	3593.6	34.375	3.08	1.21	0.57	0.2	0	0	0
7000	4240.8	86.66	2.895	0.625	0	0	0	0	0
8000	4871.4	22.175	2.45	0.36	0.09	0	0	0	0
9000	5814.3	14.275	2.005	0.625	0.235	0.15	0	0	0
10000	6122.4	24.605	3.065	0.665	0.015	0	0	0	0

4.2 译码效率仿真

本节考察 BP 算法、ML 算法以及 BPML 算法的译码效率。仿真中原始数据分组数目 k 仍分为 100～900 以及 1000～10000两段，分别以 100 和 1000 为步进值。译码器不断接收编码分组并进行译码。当恢复出所有的原始数据分组时，即停止接收，统计此时原始数据分组数目 k 与编码分组数目 P 的比值。

图 5 和图 6 分别对应于 k 在 100～900 以及 1000～10000 之间的译码效率。图中圆圈标记的曲线对应的是 BP 算法的译码效率，正方形标记的曲线对应于 ML 算法的译码效率，三角形标记的曲线对应于 BPML 算法的译码效率。可以看出，BPML 算法的译码效率与 ML 算法相近，远优于 BP 算法。

ML 算法通过求解线性方程组进行译码，利用了编码分组的所有信息，能够最大限度地恢复出原始数据分组。BPML 算法在本质上可以看作是线性方程组的快速求解。它首先寻找系数矩阵中只有一个'1'元素的所有行，构成预处理

集。假设第 i 行在预处理集当中，该行的第 j 个元素为 1，则第 j 个原始数据分组的值即为第 i 个编码分组的值；挑出所有第 j 个元素为 1 的行，将这些行与第 i 行进行异或，并将这些行所对应的编码分组的值与第 i 行编码分组的值进行异或。通过与第 i 行异或运算后，只剩一个'1'元素的行被加入到预处理集当中，将第 i 行从预处理集当中剔出。如此依次处理预处理集中的行，直到预处理集为空。如果此时仍未恢复出所有的原始数据分组，再利用剩下未处理的行求解线性方程组。BPML 算法和 ML 算法本质上都是求解线性方程组，所以两者的译码效率相近。BP 算法由于没有能够利用停止集中包含的关于原始数据分组的信息，因此译码效率最低。

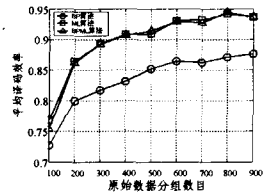

图 5 平均译码效率对比图(k：100～900)

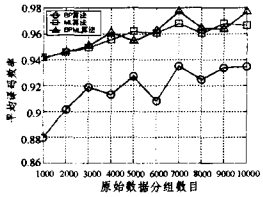

图 6 平均译码效率对比图(k：1000～10000)

4.3 译码时间仿真

与 BP 算法相比，BPML 算法虽然提高了译码效率，但在处理停止集时需要求解方程组，因此译码复杂度会高于 BP 算法。本节对 BP 算法、ML 算法和 BPML 算法的译码时间进行仿真，以此作为衡量译码复杂度的指标。原始数据分组 k 的取值与上一小节相同。对于每个 k 值，上一小节已经得出每种算法的平均译码效率 e，通过 e 求得每种算法在每个 k 值点对应的 P 值。译码器接收 P 个编码分组并进行译码，统计译码完成的时间。每种算法每个 k 值仿真 200 次，取 200 次仿真结果的均值。

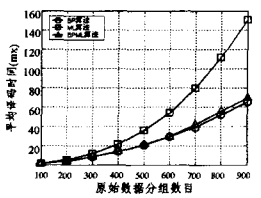

图 7 平均译码时间对比图(k：100～900)

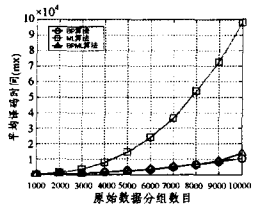

图 8 平均译码时间对比图(k：1000～10000)

原始数据分组数目 k 在 100～900 和 1000～10000 的平均译码时间如图 7 和图 8 所示。图中圆圈、正方形、三角形标记的曲线分别对应于 BP 算法、ML 算法以及 BPML 算法。从图中可以看出，BPML 算法的译码时间接近于 BP 算法的译码时间，远小于 ML 算法的译码时间。BP 算法基于简单的异或运算，运算量仅为 $O(k \cdot \ln k)$，而 ML 算法需要复杂的矩阵运算，运算量为 $O(Pk^2)$，所以 BP 算法的译码时间远小于 ML 算法。BPML 算法与 BP 算法的差异在于停止集的处理，而与 ML 算法的差异在于译码矩阵的规模。BPML 算法先采用 BP 算法进行译码，当遇到停止集无法继续译码时，再采用 ML 算法对停止集进行处理。假设 BPML 算法最终处理的停止集大小为 $N_{stopset}$，则整个译码过程的运算量约为 $O((k-N_{stopset})\ln(k-N_{stopset}))+O(N^3_{stopset})$。由前面对停止集的仿真可知，设置合理的译码开销可以保证 Nstopset 很小，因此 BPML 算法的运算复杂度仅略高于 BP 算法。单采用 ML 算法进行译码时，译码矩阵的规模要远大于 BPML 算法的译码矩阵，所以其译码时间也要远大于 BPML 算法的译码时间。

结束语 喷泉码能够在网络层上有效解决异构网络中存在的分组成功分发概率低、时延大、信道利用率低的问题。LT 码是喷泉码中一种重要的编码方式，如何提高译码效率是 LT 码研究中一直关注的问题。本文对影响 LT 码 BP 译码算法的停止集进行了理论分析和仿真，提出了消除停止集影响的 BPML 译码算法。仿真结果表明，BPML 算法结合了 BP 算法复杂度低和 ML 算法译码效率高的优点，与 BP 算法相比大大提高了译码效率。本文的研究成果同样适用于 Raptor 码的译码，对提高计算机网络中组播和广播等数据分发应用的分发效率具有实用价值。

参 考 文 献

[1] Elias P. Coding for Two Noisy Channels[C]// Proceedings of the Third London Symposium on Information Theory. London U. K. ,1955:61-76

[2] Byers J W, Luby M, Mitzenmacher M, et al. A Digital Fountain Approach to Reliable Distribution of Bulk Data[C]// Proceedings of ACM Sigcomm '98. Vancouver, Canada, Sept 1998:56-67

[3] Luby M, Mitzenmacher M, Shokrollahi A, et al. Efficient Erasure Correcting codes[J]. IEEE Transactions on Information Theory, 2001, 47(2):569-584

[4] Luby M. LT Codes[C]// Proceedings of the 43rd Annual IEEE Symposium on the Foundations of Computer Science(STOC). Vancouver, Canada, Nov. 2002:271-280

[5] Shokrollahi A. Raptor Codes [J]. IEEE Transactions on Information Theory, 2006, 52(6):2551-2567

[6] Digital Fountain Homepage[OL]. http://www.digitalfountain.com/iptv-mobile-broadcast-streaming-video-solutions.html, June 2005

[7] 朱宏鹏，张更新，谢智东. 喷泉码中 LT 码的次优度分布[J]. 应用科学学报(已录用)

[8] Luby M. Information Additive Code Generator and Decoder for Communication Systems [P]. U. S. Patent 6307487, Oct. 2001

[9] Harrelson C, Ip L, Wang Wei. Limited randomness LT codes[C]// The 41st Annual Allerton Conference on Communication, Control, and Computing. Monticello, Illinois, October, 2003

[10] Shokrollahi A. Multi - stage Code Generator and Decoder for Communication [P]. U. S. Patent 7068729, June 2006