Computation of Moments in the Trellis

Axel Heim and Vladimir Sidorenko
Institute of Telecommunications and Applied Information Theory
Ulm University
89081 Ulm, Germany
Email: {axel.heim, vladimir.sidorenko}@uni-ulm.de

Ulrich Sorger
Computer Science and Communications
University of Luxembourg
Luxembourg
Email: ulrich.sorger@uni.lu

***Abstract*— Decisions on sources with memory transmitted over independent channels can be taken by employing trellis calculations. In this paper, it is shown that for a certain class of functions their moments can be computed in the trellis, too. This is done by generalizing the forward/backward recursion known from the BCJR algorithm [1]. In analogy to the symbol probabilities, by introducing a constraint at a certain depth in the trellis we obtain symbol moments. These moments are required for an efficient implementation of the discriminated belief propagation algorithm in [2], and can furthermore be utilized to compute conditional entropies in the trellis.**

The moment computation algorithm has the same asymptotic complexity as the BCJR algorithm. It is applicable to any commutative semi-ring, thus also providing a generalization of the Viterbi algorithm [3].

I. Introduction

Trellises were introduced into the coding theory literature by Forney [4] as a means of describing the Viterbi algorithm for decoding convolutional codes. Bahl et al. [1] showed that block codes can also be described by a trellis, and Wolf [5] proposed the use of the Viterbi algorithm for trellis-based soft-decision decoding of block codes. In [6], McEliece investigated the complexity of a generalized Viterbi algorithm which allows efficient computation of flows in a code trellis. These results were further generalized in [7] and [8]. However, the calculation of flows does not fully exploit the capabilities of the trellis (representation): For a certain set of functions it is possible to calculate the moments of these functions in the trellis. These functions can be scalar or vectorial, as long as they are linear and fulfill a separability criterion.

For iterative decoding of coupled codes, the popular sum-product algorithm is used to calculate the symbol probabilities of the component codes. These probabilities are exchanged between component decoders until a stable solution is found. This iterative algorithm works very well for long 'turbo', low-density parity check (LDPC) and some other codes, obtained by concatenation of simple component codes in a special way. However, performance becomes poor when utilizing short or some good component codes.

Recently, Sorger [2] proposed a generalized decoder discriminating code words $\boldsymbol{c}$ by their correlation $\boldsymbol{c}\boldsymbol{r}^T$ or $\boldsymbol{c}\boldsymbol{w}^T$ with the received word $\boldsymbol{r}$ or a 'believed' word $\boldsymbol{w}$, respectively. Not only symbol probabilities are considered, but also the distribution of these probabilities over the correlation value. An efficient algorithm is introduced using the first two moments to approximate these distributions.

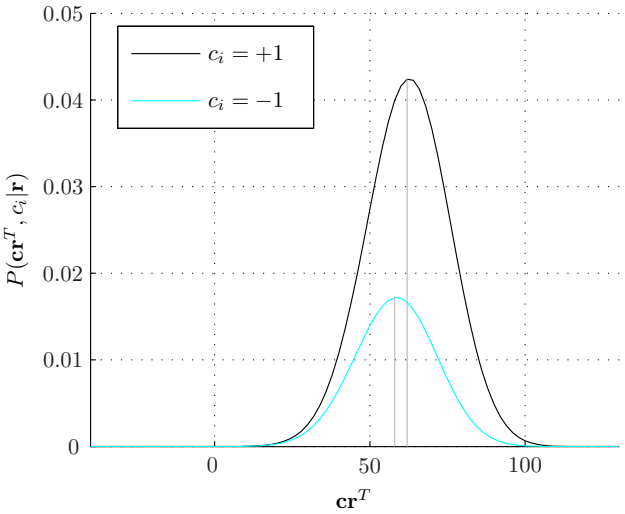

Fig. 1. Symbol Distributions of Correlation $\boldsymbol{c}\boldsymbol{r}^T$

In this paper we present an algorithm to compute such moments in the trellis.

Example 1: Consider Figure 1 which shows two distributions of the correlation function $\boldsymbol{c}\boldsymbol{r}^T$, where $\boldsymbol{c}$ is a code word and $\boldsymbol{r}$ is the noisy version of a code word $\check{\boldsymbol{c}} \in \mathbb{C}$ after transmission over a memory-less binary symmetric channel (BSC). The curves show the distributions for $\boldsymbol{c} \in \mathbb{C}_i(+1)$ and $\boldsymbol{c} \in \mathbb{C}_i(-1)$, respectively, where $\mathbb{C}_i(x) := \{\boldsymbol{c} \in \mathbb{C} : c_i = x\}$ denotes the sub-code of $\mathbb{C}$ for which the symbol c_i at a given position i of each code word equals $x \in \{-1, +1\}$. The integrals over the distributions equal the symbol probabilities $P(c_i = x|\boldsymbol{r})$. However, the probability ratio

$$\frac{P(\boldsymbol{c}\boldsymbol{r}^T, c_i = +1|\boldsymbol{r})}{P(\boldsymbol{c}\boldsymbol{r}^T, c_i = -1|\boldsymbol{r})} \tag{1}$$

varies significantly over $\boldsymbol{c}\boldsymbol{r}^T$ which can be exploited when knowledge on the correlation $\check{\boldsymbol{c}}\boldsymbol{r}^T$ with the transmitted code word is available.

The distributions in Figure 1 can be approximated with their moments

$$\mathrm{E}_{\boldsymbol{C}}\left[\left(\boldsymbol{c}\boldsymbol{r}^T\right)^m \middle| \boldsymbol{r}, c_i\right] := \sum_{\boldsymbol{c}\in\mathbb{C}} \left(\boldsymbol{c}\boldsymbol{r}^T\right)^m \cdot P(\boldsymbol{c}|\boldsymbol{r}, c_i) \tag{2}$$

up to a certain order m, where $\mathrm{E}_{\boldsymbol{C}}[.]$ is the expectation over all code words $\boldsymbol{c} \in \mathbb{C}$. The distributions will be GAUSSian for sufficiently long codes which can be understood by the law of large numbers. Hence we can expect the first two moments to suffice for a good approximation.

We present generalizations of the methods in [6] which enable us to compute expressions like $\mathrm{E}_{\boldsymbol{C}}\left[\left(\boldsymbol{c}\boldsymbol{w}^T\right)^m | \boldsymbol{r}, c_i\right]$ for some word $\boldsymbol{w}$, whereof (2) is a special case. The complexity of the algorithm is of the same order as the one of the classically used BCJR algorithm.

The remainder of this paper is structured as follows. The next section contains a review of common terminology in the context of trellises. This is extended in Section III, which deals with the computation of moments in a more general frame. In Section IV we will return to the original problem by transferring the results of Section III to linear block codes and calculate the conditional entropy in the trellis.

II. DEFINITIONS

A *trellis* $T = (\mathbb{V}, \mathbb{E})$ of rank n is a finite-directed graph[1] with vertex set $\mathbb{V}$ and edge set $\mathbb{E}$, in which every vertex is assigned a *depth* in the range $\{0, 1, \ldots, n\}$. Each edge is connecting a vertex at depth $i-1$ to one at depth i, for some $i \in \{1, 2, \ldots, n\}$. Multiple edges between vertices are allowed. The set of vertices at depth i is denoted by $\mathbb{V}_i$, so that $\mathbb{V} = \bigcup_{i=0}^{n} \mathbb{V}_i$. For $v \in \mathbb{V}_i$ we write $\mathrm{depth}(v) = i$. The set of edges connecting vertices at depth $i-1$ to those at depth i is denoted $\mathbb{E}_{i-1,i}$, so that $\mathbb{E} = \bigcup_{i=1}^{n} \mathbb{E}_{i-1,i}$. There is only one vertex at depth 0, called A, and only one at depth n, called B. If $e \in \mathbb{E}$ is a directed edge connecting the vertices u and v, which we denote by $e : u \to v$, we call u the *initial vertex*, and v the *final vertex* of e and write $\mathrm{init}(e) = u$, $\mathrm{fin(e)} = v$. We denote the number of edges leaving a vertex v by $\rho^+(v)$, and the number of edges entering a vertex v by $\rho^-(v)$, i.e.,

$$\begin{aligned} \rho^+(v) &= |\{e : \mathrm{init}(e) = v\}| \\ \rho^-(v) &= |\{e : \mathrm{fin}(e) = v\}|. \end{aligned}$$

If u and v are vertices, a *path* $\mathtt{P}$ of *length* L from u to v is a sequence of L edges: $\mathtt{P} = e_1 e_2 \cdots e_L$, such that $\mathrm{init}(e_1) = u$, $\mathrm{fin}(e_L) = v$, and $\mathrm{fin}(e_i) = \mathrm{init}(e_{i+1})$, for $i = 1, 2, \ldots, L-1$. If $\mathtt{P}$ is such a path, we sometimes write $\mathtt{P} : u \to v$ for short, as well as $\mathrm{init}(\mathtt{P}) = \mathrm{init}(e_1)$ and $\mathrm{fin}(\mathtt{P}) = \mathrm{fin}(e_L)$. We denote the set of paths from vertices at depth i to vertices at depth j by $\mathbb{E}_{i,j}$. We assume that for every vertex $v \neq A, B$, there is at least one path from A to v, and at least one path from v to B.

Example 2 (Trellis): Figure 2 shows a trellis of rank $n = 4$ with edge set $\mathbb{E} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}, \mathtt{k}, \mathtt{l}\}$ and vertex set $\mathbb{V} = \{A, 1, 2, 3, 4, 5, 6, B\}$. There are eight paths $\mathtt{P} : A \to B$ from A to B. There is $\rho^-(1) = 1$ edge entering (edge $\mathtt{a}$) and $\rho^+(1) = 2$ edges (edges $\mathtt{c}$ and $\mathtt{d}$) leaving vertex $v = 1$.

We assume each edge in the trellis is *labeled*. Let $T = (\mathbb{V}, \mathbb{E})$ be a trellis of rank n, such that each edge $e \in \mathbb{E}$ is labeled

[1]This paragraph is an excerpt from [6] with minor modifications.

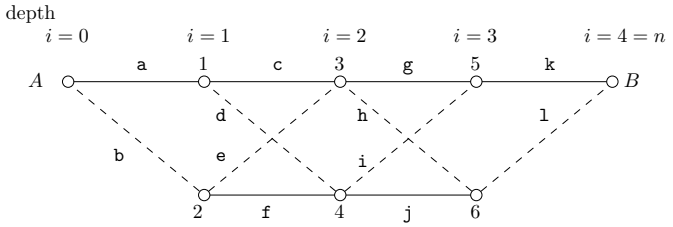

Fig. 2. Trellis of rank $n = 4$ with vertex set $\mathbb{V} = \{A, 1, 2, 3, 4, 5, 6, B\}$ and edge set $\mathbb{E} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}, \mathtt{k}, \mathtt{l}\}$

with a real valued number $\lambda(e) \in \mathbb{R}$. We now define the label of a path, and the flow between two vertices.

Definition 1 (Path Labels): The *label* $\lambda(\mathtt{P})$ of a path $\mathtt{P} = e_1 e_2 \cdots e_L$ is defined as the product $\lambda(\mathtt{P}) = \lambda(e_1) \cdot \lambda(e_2) \cdot \ldots \cdot \lambda(e_L)$ of the labels of all edges in the path. (Note that the subscript indicates the sequence number rather than the edge's depth.)

Definition 2 (Flow): If u and v are vertices in a labeled trellis, we define the *flow* $\eta(u, v)$ from u to v to be the sum of the labels on all paths from u to v, i.e.,

$$\eta(u, v) = \sum_{\mathtt{P}: u \to v} \lambda(\mathtt{P}) \,.$$

For simplicity, we only consider operations on the set of real numbers with ordinary addition and multiplication. However, the algorithm can be transferred to any commutative semi-ring, thus leading to a generalization of the Viterbi algorithm [3].

Example 3: We continue Example 2. The trellis depicted in Figure 2 is the trellis of the $(4, 3, 2)$ single parity check code. In the BCJR algorithm, the edge labels $\lambda(e)$ are the probabilities of the corresponding transitions in the channel.

III. TRELLIS-BASED COMPUTATIONS

In this section we consider distributions of the type

$$\mathcal{D} : \, q \mapsto D(q) = \sum_{\substack{\mathtt{P}: A \to B \\ f(\mathtt{P}) = q}} \lambda\left(\mathtt{P}\right)$$

for special functions f, i.e., q is mapped to the sum of the labels of all paths $\mathtt{P}$ with $f(\mathtt{P}) = q$. We present an algorithm to calculate the moments

$$\bar{\theta}^{(m)}(T) := \frac{\sum_{\mathtt{P}} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P})}{\sum_{\mathtt{P}} \lambda(\mathtt{P})}, \quad m = 0, 1, \ldots$$

in the trellis T, and - by introducing a constraint on the paths - the *symbol moments*

$$\bar{\Omega}_i^{(m)}(T, x) := \frac{\sum\limits_{\substack{\mathtt{P}: A \to B \\ c(e_i) = x}} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P})}{\sum\limits_{\substack{\mathtt{P}: A \to B \\ c(e_i) = x}} \lambda(\mathtt{P})}$$

of such distributions in the trellis. We show that the complexity of the moment calculation algorithm is $O(|\mathbb{E}|)$, where $|\mathbb{E}|$ is the number of edges in the trellis.

To each edge $e \in \mathbb{E}$ of the trellis T we introduce a second label $c(e) \in \mathbb{R}$, which we will refer to as the *c-label*. For distinction, we will call $\lambda(e)$ the *λ-label*.

Example 4: We continue Example 3. Solid lines correspond to the c-label $c(e) = 1$, dashed lines correspond to $c(e) = -1$ (bipolar binary notation). E.g., the path $\mathtt{P} = \mathtt{adik}$ has the c-label $\boldsymbol{c}(\mathtt{P}) = [+1\ -1\ -1\ +1]$ which is a code word.

Let

$$g_i\left(c(e)\right) : x \mapsto y;\ x, y \in \mathbb{R}$$

be a common function of $c(e)$ for all edges $e \in \mathbb{E}_{i-1,i}$. Further, let

$$f\left(\boldsymbol{c}(\mathtt{P})\right) = f\left(c(e_1), c(e_2), \ldots, c(e_L)\right) : \boldsymbol{c} \mapsto y;\ c(e_i), y \in \mathbb{R}$$

be a function of the c-labels of the edges of a path $\mathtt{P}$ with length L. The bold letter indicates that $\boldsymbol{c}$ is a vector. For simplicity, in the following we will abbreviate $g_i\left(c(e)\right)$ and $f\left(\boldsymbol{c}(\mathtt{P})\right)$ by $g_i(e)$ and $f(\mathtt{P})$, respectively. The functions $f(\mathtt{P})$ have to fulfill the linearity criterion

$$f(\mathtt{P}) = f(e_1 e_2 \cdots e_n) = g_1(e_1) + g_2(e_2) + \cdots + g_n(e_n) \quad (3)$$

for all paths $\mathtt{P} : A \to B$.

Definition 3 (Forward Numerator): We define the m-th *forward numerator* of a function f at vertex v of a trellis T as

$$\alpha^{(m)}(v) := \sum_{\mathtt{P}:A\to v} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P}) \quad (4)$$

with initial values

$$\alpha^{(m)}(A) := \begin{cases} 1 & : \quad m = 0 \\ 0 & : \quad m > 0 \end{cases}.$$

Theorem 1 (Forward Recursion): The m-th forward numerator $\alpha^{(m)}(v)$ of a vertex $v \in V_i$ on depth i can be recursively calculated in a trellis T by

$$\alpha^{(m)}(v) = \sum_{e:\mathrm{fin}(e)=v} \lambda(e) \cdot \sum_{l=0}^{m} \binom{m}{l} \left(g_i(e)\right)^l \cdot \alpha^{(m-l)}\left(\mathrm{init}(e)\right). \quad (5)$$

Proof: The proof is by induction on $\mathrm{depth}(v)$. For $\mathrm{depth}(v) = 1$, it follows from the definition of a trellis that all paths from A to v must consist of just one edge e, with $\mathrm{init}(e) = A$ and $\mathrm{fin}(e) = v$. Thus the true value of $\alpha^{(m)}(v)$ is the sum of the λ-labels on all edges e joining A to v, weighted by $\left(g_1(e)\right)^m$. On the other hand, the value assigned to $\alpha^{(m)}(v)$ in (5) is (because of the initialization $\alpha^{(0)}(A) = 1$, $\alpha^{(m)}(A) = 0$ for $m > 0$)

$$\alpha^{(m)}(v) = \sum_{e:A\to v} \lambda(e) \cdot \left(g_1(e)\right)^m \cdot 1$$

which is, as required, the sum of the labels on all edges e joining A to v, weighted by $\left(g_1(e)\right)^m$. Thus the algorithm works correctly for all vertices v with $\mathrm{depth}(v) = 1$ and any $m \geq 0$.

Assuming now that the assertion is true for all vertices at depth i or less and all $m \leq M$, a vertex v at depth $i+1$ is considered. Inserting the induction hypothesis in (4) into (5) we have

$$\begin{aligned}\alpha^{(m)}(v) &= \sum_{e:\mathrm{fin}(e)=v} \lambda(e) \cdot \sum_{l=0}^{m} \binom{m}{l} \left(g_i(e)\right)^l \cdot \sum_{\mathtt{P}:A\to\mathrm{init}(e)} \lambda(\mathtt{P}) \cdot \left(f(\mathtt{P})\right)^{m-l} \\ &= \sum_{e:\mathrm{fin}(e)=v} \sum_{\mathtt{P}:A\to\mathrm{init}(e)} \lambda(e)\cdot\lambda(\mathtt{P}) \cdot \sum_{l=0}^{m} \binom{m}{l} \left(g_i(e)\right)^l \cdot \left(f(\mathtt{P})\right)^{m-l};\end{aligned}$$

and using the binomial theorem we obtain

$$\alpha^{(m)}(v) = \sum_{e:\mathrm{fin}(e)=v} \sum_{\mathtt{P}:A\to\mathrm{init}(e)} \lambda(\mathtt{P}e) \cdot \left(f(\mathtt{P}) + g_i(e)\right)^m. \quad (6)$$

But every path from A to v must be of the form $\mathtt{P}e$, where $\mathtt{P}$ is a path from A to a vertex u with $\mathrm{depth}(u) = i$, $\mathrm{init}(e) = u$ and $\mathrm{fin}(e) = v$. Thus by (6) and (3), $\alpha^{(m)}(v)$ is correctly calculated by the algorithm. ∎

Remark 1 (Flow): $\alpha^{(0)}(v)$ in (4) is the flow $\eta(A, v)$ from A to v (cf. Definition 2) as it is calculated during the forward recursion of the BCJR algorithm.

Theorem 2 (Complexity): The computation of the forward numerators up to order M for all vertices in a trellis requires $O(|\mathbb{E}|)$ arithmetic operations, i.e., multiplications and additions.

Proof: The sum over l in (5) requires m additions and $(m+1) \cdot 2$ multiplications (disregarding the computation of $\left(g_i(e)\right)^l$), the sum over e requires $(\rho^-(v) - 1) + \rho^-(v) \cdot m$ additions and $\rho^-(v) \cdot (1 + (m+1) \cdot 2)$ multiplications. Hence, for a vertex $v \in V_i$,

$$\sum_{m=0}^{M} \left(\rho^-(v) \cdot (m+1) - 1\right) = \rho^-(v) \cdot \left(\frac{1}{2}M^2 + \frac{3}{2}M + 1\right) - (M+1)$$

additions and

$$\sum_{m=0}^{M} \rho^-(v) \cdot (2m+3) = \rho^-(v) \cdot \left(M^2 + 4M + 3\right)$$

multiplications are necessary. Summing over all vertices except A, and with $|\mathbb{E}| = \sum_{i=1}^{n} \sum_{v\in V_i} \rho^-(v)$ the requirement of additions and multiplications is

$$\begin{aligned}\mathtt{add} &= \left(\frac{1}{2}M^2 + \frac{3}{2}M + 1\right) \cdot |\mathbb{E}| - (M+1) \cdot (|\mathbb{V}| - 1) \\ \mathtt{mult} &= (M^2 + 4M + 3) \cdot |\mathbb{E}|.\end{aligned}$$

With $|\mathbb{V}| \geq 1$ the total number of operations is thus bounded above by $\left(\frac{3}{2}M^2 + \frac{11}{2}M + 4\right) \cdot |\mathbb{E}|$. ∎

In analogy to the forward numerator in Definition 3 we can also define a backward numerator.

Definition 4 (Backward Numerator): The m-th *backward numerator* of a vertex $v \in \mathbb{V}_i$ is defined as

$$\beta^{(m)}(v) := \sum_{\mathtt{P}:v\to B} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P})$$

with initial values

$$\beta^{(m)}(B) = \begin{cases} 1 & : \quad m = 0 \\ 0 & : \quad m > 0 \end{cases}.$$

Theorem 3 (Backward Recursion): The m-th backward numerator $\beta^{(m)}(v)$ of a vertex $v \in \mathbb{V}_i$ can be calculated in a trellis T by

$$\beta^{(m)}(v) = \sum_{e:\mathrm{init}(e)=v} \lambda(e) \cdot \sum_{l=0}^{m} \binom{m}{l} \left(g_{i+1}(e)\right)^l \cdot \beta^{(m-l)}\left(\mathrm{fin}(e)\right).$$

Proof: The proof is analog to the proof of Theorem 1. ∎

It obviously holds that $\alpha^{(m)}(B) = \beta^{(m)}(A) =: \theta^{(m)}(T)$, providing the *m*-th *moment*

$$\bar{\theta}^{(m)}(T) := \frac{\theta^{(m)}(T)}{\theta^{(0)}(T)} = \frac{\sum\limits_{\mathtt{P}:A\to B} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P})}{\sum\limits_{\mathtt{P}:A\to B} \lambda(\mathtt{P})}$$

of the distribution of function f given T.

In analogy to the BCJR algorithm [1] for calculating symbol probabilities, we next consider the calculation of moments of f introducing a constraint on the value of the c-labels at a certain depth i in the trellis. I.e., the moments are calculated in a sub-trellis of T.

Definition 5 (Symbol Moment): We define the m-th *symbol moment* $\bar{\Omega}_i^{(m)}(T,x)$ at depth i of a trellis T as

$$\bar{\Omega}_i^{(m)}(T,x) := \frac{\sum\limits_{\substack{\mathtt{P}:A\to B \\ c_i=x}} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P})}{\sum\limits_{\substack{\mathtt{P}:A\to B \\ c_i=x}} \lambda(\mathtt{P})}$$

where $c_i = c(e_i)$ and $e_i \in \mathbb{E}_{i-1,i}$ is the i-th edge of path $\mathtt{P}$.

Theorem 4: The m-th symbol moment can be calculated by

$$\bar{\Omega}_i^{(m)}(T,x) = \frac{\Omega_i^{(m)}(T,x)}{\Omega_i^{(0)}(T,x)}$$

with

$$\Omega_i^{(m)}(T,x) = \sum_{\substack{e\in\mathbb{E}_{i-1,i}: \\ c(e)=x}} \lambda(e) \cdot \sum_{l=0}^{m} \binom{m}{l} \beta^{(m-l)}(\mathrm{fin}(e)) \cdot \cdot \sum_{k=0}^{l} \binom{l}{k} \left(g_i(e)\right)^k \cdot \alpha^{(l-k)}(\mathrm{init}(e)). \quad (7)$$

Proof: Let $\mathtt{P}_H$ and $\mathtt{P}_T$ denote the head and tail parts of the paths $\mathtt{P} : A \to B$ through the trellis T, with an edge e in between, i.e., $\mathtt{P} = \mathtt{P}_H e \mathtt{P}_T$ with $\mathrm{init}(\mathtt{P}_H) = A$, $\mathrm{fin}(\mathtt{P}_H) = \mathrm{init}(e)$, $\mathrm{fin}(e) = \mathrm{init}(\mathtt{P}_T)$ and $\mathrm{fin}(\mathtt{P}_T) = B$, for a given depth i and $e \in \mathbb{E}_{i-1,i}$. Then we can write

$$\begin{aligned}\Omega_i^{(m)}(T,x) &= \sum_{\substack{\mathtt{P}:A\to B \\ c_i=x}} \left(f(\mathtt{P})\right)^m \cdot \lambda(\mathtt{P}) \\ &= \sum_{\substack{e\in\mathbb{E}_{i-1,i}: \\ c(e)=x}} \sum_{\substack{\mathtt{P}_H:A\to \\ \mathrm{init}(e)}} \sum_{\substack{\mathtt{P}_T:\mathrm{fin}(e) \\ \to B}} \left(f(\mathtt{P}_H)+g_i(e)+f(\mathtt{P}_T)\right)^m \cdot \\ &\qquad \cdot \lambda\left(\mathtt{P}_H\, e\, \mathtt{P}_T\right).\end{aligned}$$

Applying the binomial theorem twice and separating the λ-labels we obtain

$$\begin{aligned}\Omega_i^{(m)}(T,x) = \sum_{\substack{e\in\mathbb{E}_{i-1,i}: \\ c(e)=x}} \lambda(e) \cdot \sum_{l=0}^{m} \binom{m}{l} \sum_{\substack{\mathtt{P}_T: \\ \mathrm{fin}(e)\to B}} \left(f(\mathtt{P}_T)\right)^{m-l} \cdot \lambda(\mathtt{P}_T) \cdot \\ \cdot \sum_{k=0}^{l} \binom{l}{k} \left(g_i(e)\right)^k \cdot \sum_{\substack{\mathtt{P}_H: \\ A\to \mathrm{init}(e)}} \left(f(\mathtt{P}_H)\right)^{l-k} \cdot \lambda(\mathtt{P}_H),\end{aligned}$$

and using the definitions of forward and backward numerators finally yields the assertion of the theorem. ∎

Remark 2 (Forward/Backward Moments): For numeric reasons it may be advantageous to directly compute the *forward* and *backward moments*

$$\bar{\alpha}^{(m)}(v) := \frac{\alpha^{(m)}(v)}{\alpha^{(0)}(v)} \quad \text{and} \quad \bar{\beta}^{(m)}(v) := \frac{\beta^{(m)}(v)}{\beta^{(0)}(v)},$$

respectively, and to calculate and carry the 0-th numerators (flows) in the logarithmic domain.

Remark 3: We cannot only determine the moments of a trellis or sub-trellis, but also of a single edge.

Remark 4: The symbol distribution for two sub-trellises of the $[7\ 5]_{\mathrm{oct}}$ convolutional code, namely the sub-codes with the i-th code bit $c_i = +1$ and $c_i = -1$, respectively, is given in Example 1. The curves obtained by GAUSSian approximation almost coincide with the ones plotted in Figure 1.

Remark 5: It is straight forward to extend the proposed algorithm to the calculation of joint moments of two or more functions.

IV. APPLICATIONS

We will now apply the results of Section III to linear block codes. We show how to compute the moments

$$\mathrm{E}_{\boldsymbol{C}}\left[\left(H(\boldsymbol{c}|\boldsymbol{w})\right)^m |\boldsymbol{r}, c_i = x\right] := \sum_{\boldsymbol{c}\in\mathbb{C}} \left(H(\boldsymbol{c}|\boldsymbol{w})\right)^m P(\boldsymbol{c}|\boldsymbol{r}, c_i = x) \quad (8)$$

of the distribution

$$\mathcal{D} : q = H(\boldsymbol{c}|\boldsymbol{w}) \mapsto P\left(q|\boldsymbol{r}, c_i = x\right) = \sum_{\substack{\boldsymbol{c}\in\mathbb{C}: \\ H(\boldsymbol{c}|\boldsymbol{w})=q}} P\left(\boldsymbol{c}|\boldsymbol{r}, c_i = x\right)$$

over all code words $\boldsymbol{c} \in \mathbb{C}$ given a received word $\boldsymbol{r}$ and the i-th code bit being $c_i = x \in \{-1, 1\}$, where $P(\boldsymbol{c}|\boldsymbol{r})$ is the conditional probability of $\boldsymbol{c}$ given $\boldsymbol{r}$. However, both for soft decision (AWGN channel) and hard decision (binary symmetric channel) with equiprobable code words the conditional uncertainty

$$H(\boldsymbol{c}|\boldsymbol{w}) = K_1 + K_2 \cdot \boldsymbol{c}\boldsymbol{w}^T \quad (9)$$

of $\boldsymbol{c}$ given a word $\boldsymbol{w}$ linearly relates to the correlation with constants K_1 and K_2. Thus we can equivalently compute the moments

$$\mathrm{E}_{\boldsymbol{C}}\left[\left(\boldsymbol{c}\boldsymbol{w}^T\right)^m |\boldsymbol{r}, c_i = x\right] = \sum_{\boldsymbol{c}\in\mathbb{C}} \left(\boldsymbol{c}\boldsymbol{w}^T\right)^m \cdot P(\boldsymbol{c}|\boldsymbol{r}, c_i = x) \quad (10)$$

of $\boldsymbol{c}\boldsymbol{w}^T$ in the trellis and afterwards apply the binomial theorem to obtain (8).

These moments are required, e.g., for the discriminated belief propagation algorithm in [2]. As a special case we can calculate the conditional mean uncertainty or *entropy*

$$H(\boldsymbol{C}|\boldsymbol{r}) = \sum_{\boldsymbol{c}\in\mathbb{C}} H(\boldsymbol{c}|\boldsymbol{r}) \cdot P(\boldsymbol{c}|\boldsymbol{r})$$

of a code or sub-code given $\boldsymbol{r}$.

Consider a binary linear block code $\mathbb{C}$ of length n which is representable in a trellis, e.g., a terminated convolutional code. Let the c-labels $c(e) = c_i \in \{\pm 1\}$ be the bipolar representation of the code bit labeling edge $e \in \mathbb{E}_{i-1,i}$. To each path $\mathtt{P} : A \to B$ it belongs a sequence $\boldsymbol{c}(\mathtt{P})$ of n c-labels representing a code word $\boldsymbol{c} \in \mathbb{C}$. Let $\boldsymbol{r} = [r_1 r_2 \cdots r_n]$, $r_i \in \mathbb{R}$, be the noisy version of a code word $\boldsymbol{c}$ after transmission over a memory-less channel. Let the λ-label of a path $\mathtt{P}$ be the conditional probability of the received word $\boldsymbol{r}$ given the code word $\boldsymbol{c}$, i.e., $\lambda(\mathtt{P}) = P(\boldsymbol{r}|\boldsymbol{c})$. Let further the function f of the paths' c-labels, i.e., the function of the code words, be the correlation (inner product) of $\boldsymbol{w}$ and $\boldsymbol{c}$,

$$f(\mathtt{P}) = f(\boldsymbol{c}(\mathtt{P})) = \boldsymbol{c}\boldsymbol{w}^T = \sum_{i=1}^{n} c_i w_i \,.$$

Hence, $g_i(e_i) = c_i w_i$ and the separability criterion (3) is fulfilled. Each path in the trellis of $\mathbb{C}$ uniquely maps to a code word, and we can apply the theorems of Section III replacing $\sum_{\mathtt{P}}$ by $\sum_{\boldsymbol{c}}$. Applying BAYES' rule to (10),

$$\mathrm{E}_{\boldsymbol{C}}\left[\left(\boldsymbol{c}\boldsymbol{w}^T\right)^m |\boldsymbol{r}, c_i = x\right] = \frac{\sum\limits_{\boldsymbol{c}\in\mathbb{C}:c_i=x} \left(\boldsymbol{c}\boldsymbol{w}^T\right)^m P(\boldsymbol{r}|\boldsymbol{c})}{\sum\limits_{\boldsymbol{c}\in\mathbb{C}:c_i=x} P(\boldsymbol{r}|\boldsymbol{c})}, \quad (11)$$

and comparing with Definition 5 we observe that Theorems 1 and 3 hold, and hence these moments can be calculated in the trellis according to Theorem 4 as the symbol moments

$$\mathrm{E}_{\boldsymbol{C}}\left[\left(\boldsymbol{c}\boldsymbol{w}^T\right)^m |\boldsymbol{r}, c_i = x\right] = \bar{\Omega}_i^{(m)}(x).$$

Analogously, when omitting the code bit constraint $c_i = x$, the moments are given by

$$\mathrm{E}_{\boldsymbol{C}}\left[\left(\boldsymbol{c}\boldsymbol{w}^T\right)^m |\boldsymbol{r}\right] = \sum_{\boldsymbol{c}\in\mathbb{C}} \left(\boldsymbol{c}\boldsymbol{w}^T\right)^m \cdot P(\boldsymbol{c}|\boldsymbol{r}) = \bar{\theta}^{(m)}(T).$$

For $\boldsymbol{w} = \boldsymbol{r}$, $m = 1$ and $g_i(e) = c_i r_i$ we can thus calculate the conditional entropies

$$H(\boldsymbol{C}|\boldsymbol{r}) = \sum_{\boldsymbol{c}\in\mathbb{C}} H(\boldsymbol{c}|\boldsymbol{r}) \cdot P(\boldsymbol{c}|\boldsymbol{r}) = K_1 + K_2 \cdot \bar{\theta}^{(1)}(T)$$

and

$$H(\boldsymbol{C}_i(x)|\boldsymbol{r}) = \sum_{\boldsymbol{c}\in\mathbb{C}:c_i=x} H(\boldsymbol{c}|\boldsymbol{r}) \cdot P(\boldsymbol{c}|\boldsymbol{r}) = K_1 + K_2 \cdot \bar{\Omega}_i^{(1)}(x)$$

of the code $\mathbb{C}$ and the sub-code $\mathbb{C}_i(x) = \{\boldsymbol{c} \in \mathbb{C} : c_i = x\}$ given $\boldsymbol{r}$, respectively. While $H(\boldsymbol{C}|\boldsymbol{r})$ can also be calculated with the classical BCJR algorithm as

$$\begin{aligned}\sum_{\boldsymbol{c}\in\mathbb{C}} \boldsymbol{c}\boldsymbol{r}^T \cdot P(\boldsymbol{c}|\boldsymbol{r}) &= \sum_{i=1}^{n} \sum_{\boldsymbol{c}\in\mathbb{C}} c_i r_i \cdot P(\boldsymbol{c}|\boldsymbol{r}) \\ &= \sum_{i=1}^{n} r_i \cdot \left(\sum_{\substack{\boldsymbol{c}\in\mathbb{C}:\\ c_i=1}} P(\boldsymbol{c}|\boldsymbol{r}) - \sum_{\substack{\boldsymbol{c}\in\mathbb{C}:\\ c_i=-1}} P(\boldsymbol{c}|\boldsymbol{r}) \right),\end{aligned}$$

this does not hold for the conditional entropy of $\mathbb{C}_i(x)$.

V. CONCLUSIONS

A trellis represents a general distribution which can be marginalized, e.g., with respect to edge labels. An algorithm for the computation of moments in the trellis was presented. It was derived by generalizing the forward/backward recursion known from the BCJR algorithm. The results were transferred to the concrete problem of computing the moments of the correlation between a block code and some given word. The algorithm is a requirement for efficient implementation of the discriminated belief propagation algorithm in [2]. It can also be used to calculate the conditional entropy of a code or sub-code. The asymptotic complexity of the algorithm is the same as for the BCJR algorithm.

REFERENCES

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, pp. 284 – 287, March 1974.

[2] U. Sorger, "Discriminated belief propagation," October 2007. http://arxiv.org/abs/0710.5501.

[3] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, pp. 260 – 269, April 1967.

[4] G. Forney, Jr., "Review of random tree codes," Appendix A of Final Report on Contract NAS2-3637, NASA CR73176, NASA Ames Res. Ctr., CA, Dec. 1967.

[5] J. Wolf, "Efficient maximum-likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. 24, pp. 76–80, Jan. 1978.

[6] R. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inf. Theory*, vol. 42, pp. 1072 – 1092, July 1996.

[7] S. Aji and R. McEliece, "The generalized distributive law," *IEEE Trans. Inf. Theory*, vol. 46, pp. 325–343, March 2000.

[8] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.