On the applications of factor graphs and the sum-product algorithm to detection and decoding

Giulio Colavolpe

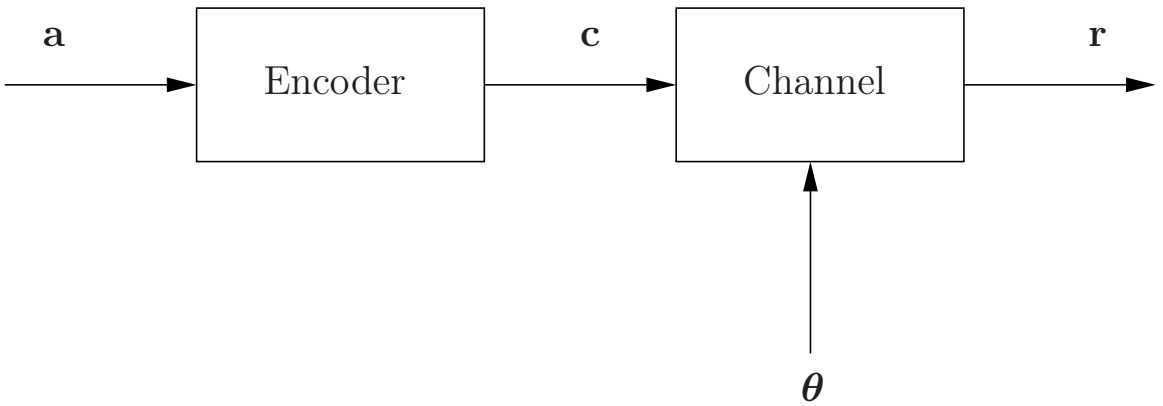
SPADiC Lab &
CNIT Research Unit
Dipartimento di
Ingegneria dell'Informazione
Università of Parma
Parma, Italy

Outline

Outline

Introduction

Considered system (at the receiver side, by means of a discretization process, the received signal $r(t)$ is converted into a discrete-time sequence $\mathbf{r}$)



$$\mathbf{a} = (a_1, a_2, \dots)$$
$$\mathbf{c} = (c_1, c_2, \dots)$$
$$\boldsymbol{\theta} = (\theta_1, \theta_2, \dots)$$
$$\mathbf{r} = (r_1, r_2, \dots)$$

Problems of interest in the present talk:

- MAP symbol detection:

$$\hat{a}_k = \underset{a_k}{\operatorname{argmax}}\, P(a_k|\mathbf{r})$$

- MAP sequence detection:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmax}}\, P(\mathbf{a}|\mathbf{r})$$

- MAP estimation:

$$\hat{\theta}_k = \underset{\theta_k}{\operatorname{argmax}}\, p(\theta_k|\mathbf{r})$$

Introduction (cont'd)

- All problems have, in general, exponential complexity in the transmission length (NP-hard), unless proper conditions occur
- The first and the third problems have in common a marginalization of the joint distribution $P(\mathbf{a}, \mathbf{c}, \theta|\mathbf{r})$
- Apparently, this is not the case of the second problem. We will come back on MAP sequence detection later
- When is it possible to implement a marginalization in a more effective way?

Introduction (cont'd)

- Algorithms that must deal with complicated global functions of many variables often exploit the manner in which the given functions factor as a product of “local" functions, each of which depends on a subset of the variables
- Such a factorization can be visualized with a factor graph, a bipartite graph that expresses which variables are arguments of which local functions
- The sum-product algorithm works on the factor graph and computes—either exactly or approximately—the marginal functions derived from the global function
- A wide variety of algorithms developed in artificial intelligence, signal processing and digital communications can be derived as specific instances of the sum-product algorithm

Outline

Definition

- Let $x_1, x_2, \ldots, x_n$ be a collection of variables
- x_i takes on values in some (usually finite) domain (or alphabet) A_i
- Let $g(x_1, x_2, \ldots, x_n)$ be an $\mathbb{R}$-valued function of these variables, i.e., a function with domain $S = A_1 \times A_2 \times \cdots \times A_n$ and codomain $\mathbb{R}$
- Associated with $g(x_1, x_2, \ldots, x_n)$ there are n marginal functions $g_i(x_i)$ defined as

$$g_i(x_i) = \sum_{x_1 \in A_1} \cdots \sum_{x_{i-1} \in A_{i-1}} \sum_{x_{i+1} \in A_{i+1}} \cdots \sum_{x_n \in A_n} g(x_1, x_2, \ldots, x_n)$$

- This operation is called not-sum or summary and will be denoted by

$$g_i(x_i) = \sum_{\sim\{x_i\}} g(x_1, x_2, \ldots, x_n)$$

Factor Graphs (FGs)

- Suppose that $g(x_1, x_2, \ldots, x_n)$ factors into a product of several local functions, each having some subset of $\{x_1, x_2, \ldots, x_n\}$ as arguments:

$$g(x_1, x_2, \ldots, x_n) = \prod_{j \in J} f_j(X_j)$$

  where J is a discrete index set, X_j is a subset of $\{x_1, x_2, \ldots, x_n\}$, and $f_j(X_j)$ is a function having the elements of X_j as arguments

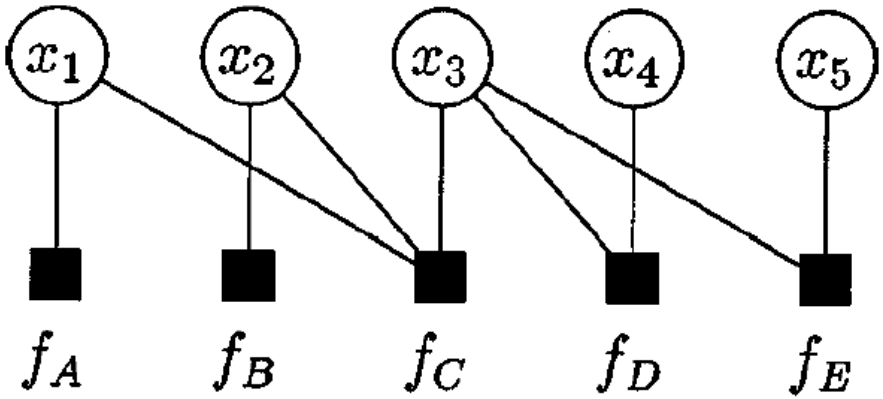
- A factor graph is a bipartite graph that expresses the structure of this factorization
- It has a variable node for each variable x_i, a factor node for each local function f_j, and an edge connecting variable node x_i to function node f_j if and only if x_i is an argument of f_j

Example

- Let $g(x_1, x_2, \ldots, x_n)$ be a function of five variables, and suppose that g can be expressed as a product

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5)$$

- In this case, $J = \{A, B, C, D, E\}$, $X_A = \{x_1\}$, $X_B = \{x_2\}$, $X_C = \{x_1, x_2, x_3\}$, $X_D = \{x_3, x_4\}$, and $X_E = \{x_3, x_5\}$



Marginalization of a Joint Pmf

- If $g(x_1, x_2, \ldots, x_n)$ is a joint probability mass function (PMF), we are interested in computing the marginal functions $g_i(x_i)$
- In the case of the previous example, by using the distributive law:

$$g_1(x_1) = f_A(x_1)\Big\{\sum_{x_2} f_B(x_2)\Big[\sum_{x_3} f_C(x_1, x_2, x_3)\Big(\sum_{x_4} f_D(x_3, x_4)\Big)\Big(\sum_{x_5} f_E(x_3, x_5)\Big)\Big]\Big\}$$

  ($a(b + c)$ is more cost-effective than $ab + ac$)
- In summary notation

$$g_1(x_1) = f_A(x_1) \sum_{\sim\{x_1\}} \Big\{f_B(x_2) f_C(x_1, x_2, x_3)\Big(\sum_{\sim\{x_3\}} f_D(x_3, x_4)\Big)\Big(\sum_{\sim\{x_3\}} f_E(x_3, x_5)\Big)\Big\}$$

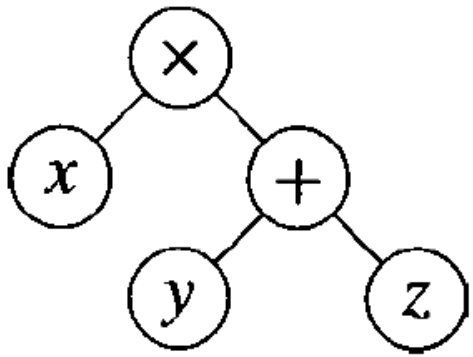
- Similarly

$$g_3(x_3) = \Big[\sum_{\sim\{x_3\}} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3)\Big]\Big[\sum_{\sim\{x_3\}} f_D(x_3, x_4)\Big]\Big[\sum_{\sim\{x_3\}} f_E(x_3, x_5)\Big]$$

- We have some operations in common!

Expression Trees

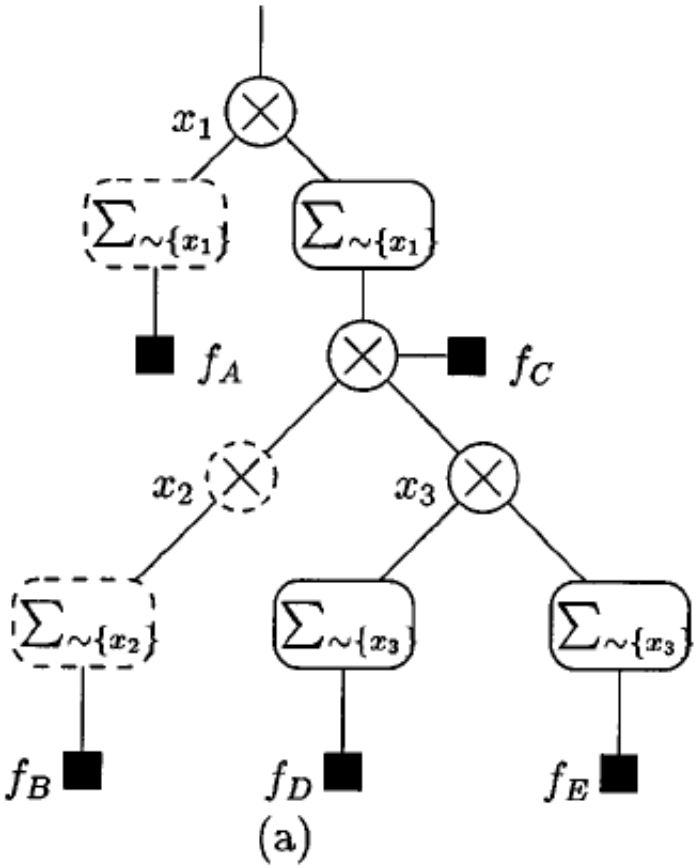
- In computer science, arithmetic expressions are often represented by ordered rooted trees (the expression trees) in which internal vertices (i.e., vertices with descendants) represent arithmetic operators and leaf vertices (i.e., vertices without descendants) represent variables or constants. Ex.: $x(y+z)$



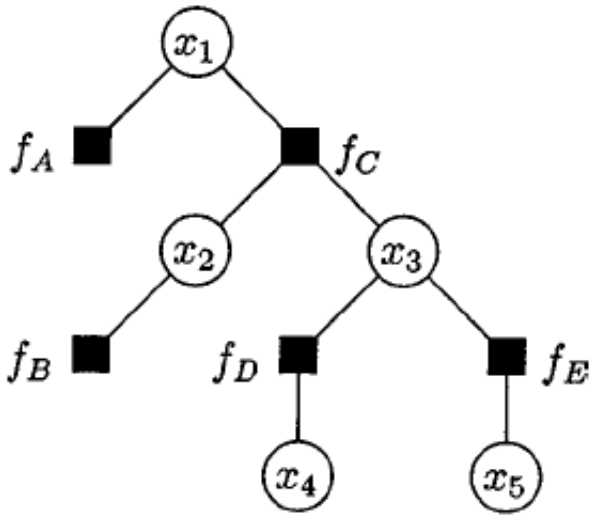
- Expression trees may be extended such that the leaf vertices represent functions not just variables or constants

Expression Trees: $g_1(x_1)$

$$g_1(x_1) = f_A(x_1) \sum_{\sim\{x_1\}} \Big\{ f_B(x_2) f_C(x_1, x_2, x_3) \Big(\sum_{\sim\{x_3\}} f_D(x_3, x_4) \Big) \Big(\sum_{\sim\{x_3\}} f_E(x_3, x_5) \Big) \Big\}$$



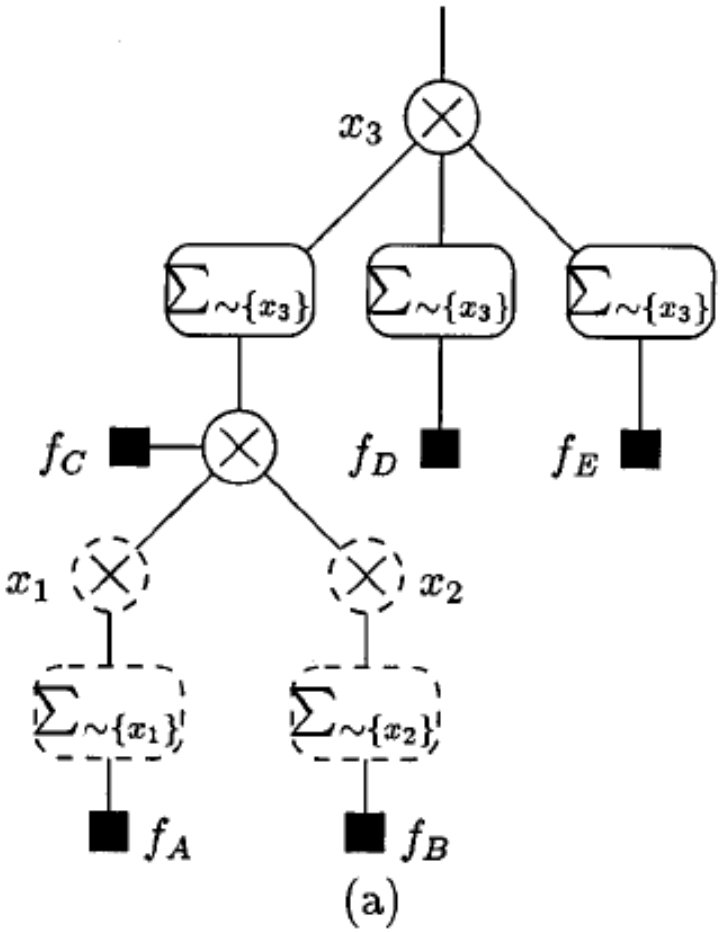
(a)



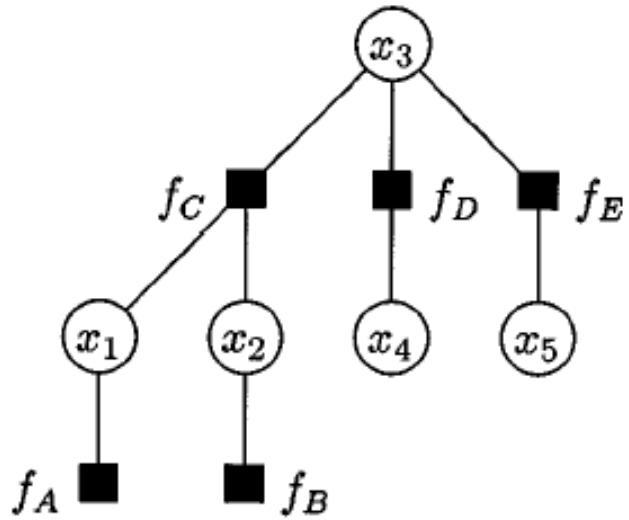
(b)

Expression Trees: $g_3(x_3)$

$$g_3(x_3) = \Big[\sum_{\sim\{x_3\}} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3)\Big]\Big[\sum_{\sim\{x_3\}} f_D(x_3, x_4)\Big]\Big[\sum_{\sim\{x_3\}} f_E(x_3, x_5)\Big]$$



(a)



(b)

Expression Trees and Factor Graphs

- When a factor graph is cycle free, the factor graph not only encodes in its structure the factorization of the global function, but also encodes arithmetic expressions by which the marginal functions associated with the global function may be computed, following these rules
  - Replace each variable node in the factor graph with a product operator
  - Replace each factor node with a "form product and multiply by f" operator
  - Between a factor node f and its parent x insert a $\sum_{\sim\{x\}}$ summary operator
- Trivial products (those with one or no operand) act as identity operators
- A summary operation $\sum_{\sim\{x\}}$ applied to a function with a single argument x is also a trivial operation, and may be omitted

Computing a Single Marginal Function

- Every expression tree represents an algorithm for computing the corresponding expression, a “bottom-up” procedure that begins at the leaves of the tree, with each operator vertex combining its operands and passing on the result as an operand for its parent
- Rather than working with the expression tree, it is simpler and more direct to describe such marginalization algorithms in terms of the corresponding factor graph
- Let us imagine that there is a processor associated with each vertex of the factor graph and that the factor-graph edges represent channels by which these processors may communicate

Single-i Sum-Product Algorithm

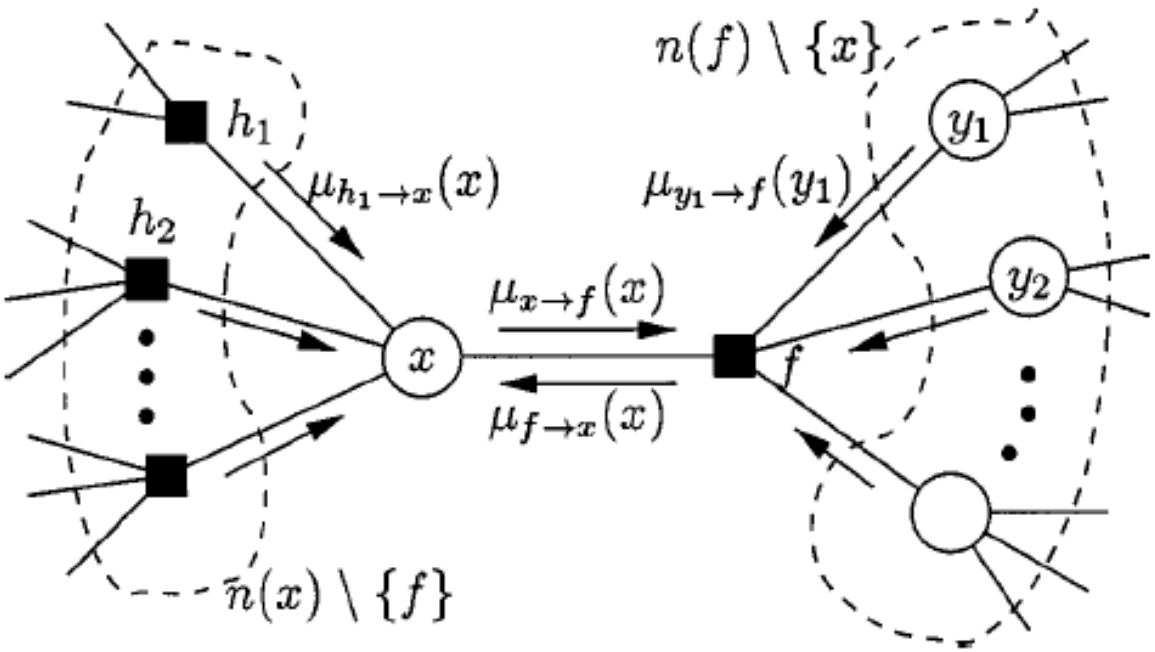
- It computes a single marginal function $g_i(x_i)$ in a rooted cycle-free factor graph, with x_i taken as root vertex
- Rules
  - Each leaf variable node sends a trivial “identity function" message to its parent
  - Each leaf factor node f sends a description of f to its parent
  - Each vertex waits for messages from all of its children before computing the message to be sent to its parent
  - A variable node simply sends the product of messages received from its children
  - A factor node f with parent x forms the product of f with the messages received from its children, and then operates on the result with a $\sum_{\sim\{x\}}$ summary operator
  - The computation terminates at the root node x_i, where the marginal function $g_i(x_i)$ is obtained as the product of all messages received at x_i

Computing All Marginal Functions

- Computation of $g_i(x_i)$ for all i simultaneously can be efficiently accomplished by essentially overlaying on a single factor graph all possible instances of the single-i algorithm
- No particular vertex is taken as a root vertex, so there is no fixed parent/child relationship among neighboring vertices
- Message passing is initiated at the leaves
- Each vertex v remains idle until messages have arrived on all but one of the edges incident on v; then, it is able to compute the message on the remaining edge to another vertex w as in the single-i algorithms. After that, the vertex v returns to the idle state, waiting for a return message from w. Once this message has arrived, the vertex is able to compute and send messages to each of its neighbors (other than w)

Computing All Marginal Functions (cont'd)

- The algorithm terminates once two messages have been passed over every edge, one in each direction
- At the variable node x_i, the product of all incoming messages is the marginal function $g_i(x_i)$, just as in the single-i algorithm



Sum-Product Algorithm

- The resulting algorithm is called sum-product algorithm and is based on the following two rules. Let us denote by $n(v)$ the set of neighbors of a given node v
  - variable to local function

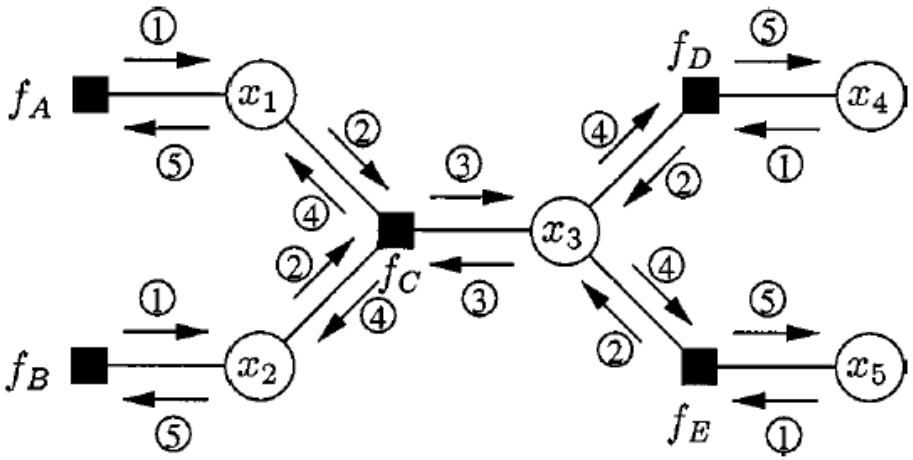
$$\mu_{x\to f}(x) = \prod_{h\in n(x)\setminus\{f\}} \mu_{h\to x}(x)$$

  - local function to variable

$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} \left(f(X) \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \right)$$

    where $X = n(f)$ is the set of arguments of the function f
- Note that all messages are functions

Example



- Step 1

$$\mu_{f_A \to x_1}(x_1) = \sum_{\sim\{x_1\}} f_A(x_1) = f_A(x_1) \qquad \mu_{f_B \to x_2}(x_2) = \sum_{\sim\{x_2\}} f_B(x_2) = f_B(x_2)$$

$$\mu_{x_4 \to f_D}(x_4) = 1 \qquad \mu_{x_5 \to f_E}(x_5) = 1$$

- Step 2

$$\mu_{x_1 \to f_C}(x_1) = \mu_{f_A \to x_1}(x_1) \qquad \mu_{x_2 \to f_C}(x_2) = \mu_{f_B \to x_2}(x_2)$$

$$\mu_{f_D \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_4 \to f_D}(x_4) f_D(x_3, x_4)$$

Example (cont'd)

- Step 3

$$\mu_{f_C \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_1 \to f_C}(x_1)\mu_{x_2 \to f_C}(x_2) f_C(x_1, x_2, x_3)$$

$$\mu_{x_3 \to f_C}(x_3) = \mu_{f_D \to x_3}(x_3)\mu_{f_E \to x_3}(x_3)$$

- Step 4

$$\mu_{f_C \to x_1}(x_1) = \sum_{\sim\{x_1\}} \mu_{x_3 \to f_C}(x_3)\mu_{x_2 \to f_C}(x_2) f_C(x_1, x_2, x_3)$$

$$\mu_{f_C \to x_2}(x_2) = \sum_{\sim\{x_2\}} \mu_{x_3 \to f_C}(x_3)\mu_{x_1 \to f_C}(x_1) f_C(x_1, x_2, x_3)$$

$$\mu_{x_3 \to f_D}(x_3) = \mu_{f_C \to x_3}(x_3)\mu_{f_E \to x_3}(x_3)$$

$$\mu_{x_3 \to f_E}(x_3) = \mu_{f_C \to x_3}(x_3)\mu_{f_D \to x_3}(x_3)$$

- Step 5

$$\mu_{x_1 \to f_A}(x_1) = \mu_{f_C \to x_1}(x_1)$$

$$\mu_{x_2 \to f_B}(x_2) = \mu_{f_C \to x_2}(x_2)$$

$$\mu_{f_D \to x_4}(x_4) = \sum_{\sim\{x_4\}} \mu_{x_3 \to f_D}(x_3) f_D(x_3, x_4)$$

Example (cont'd)

- Termination

$$
\begin{aligned}
g_1(x_1) &= \mu_{f_A \to x_1}(x_1)\mu_{f_C \to x_1}(x_1) \\
g_2(x_2) &= \mu_{f_B \to x_2}(x_2)\mu_{f_C \to x_2}(x_2) \\
g_3(x_3) &= \mu_{f_C \to x_3}(x_3)\mu_{f_D \to x_3}(x_3)\mu_{f_E \to x_3}(x_3) \\
g_4(x_4) &= \mu_{f_D \to x_4}(x_4) \\
g_5(x_5) &= \mu_{f_E \to x_5}(x_5)
\end{aligned}
$$

- Equivalently

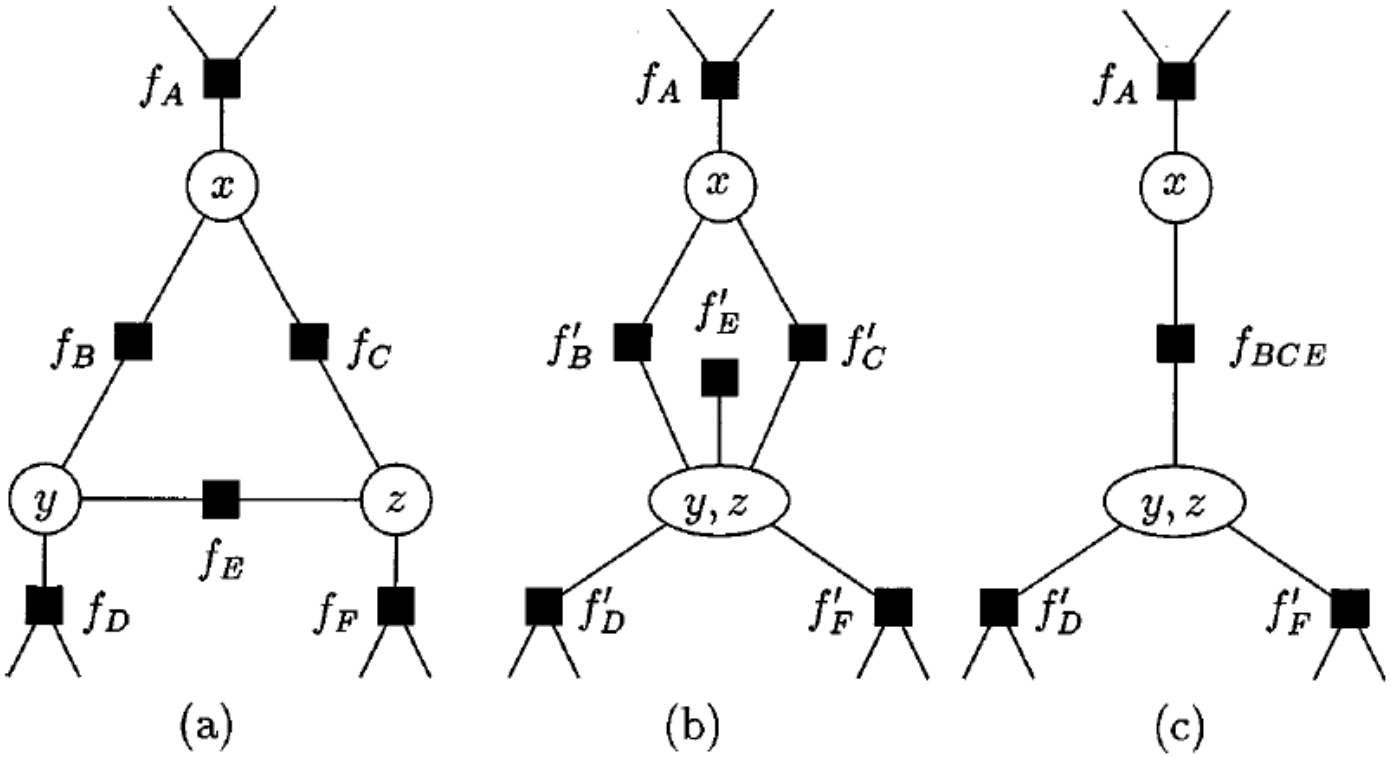
$$
\begin{aligned}
g_3(x_3) &= \mu_{f_C \to x_3}(x_3)\mu_{x_3 \to f_C}(x_3) \\
&= \mu_{f_D \to x_3}(x_3)\mu_{x_3 \to f_D}(x_3) \\
&= \mu_{f_E \to x_3}(x_3)\mu_{x_3 \to f_E}(x_3)
\end{aligned}
$$

Marginalization in Graphs with Cycles

- When a graph has cycles, in the sum-product algorithm there is not a natural termination
- Because of the cycles in the graph, an iterative algorithm with no natural termination will result, with messages passed multiple times on a given edge
- It is not possible to obtain an exact marginalization of the global function
- Some of the most exciting applications of the sum-product algorithm (turbo codes, LDPC codes) arise precisely in situations in which the underlying factor graph does have cycles
- Extensive simulation results show that such decoding algorithms can achieve astonishing performance even though the underlying factor graph has cycles
- The adopted schedule assumes a key role (ex. flooding schedule)

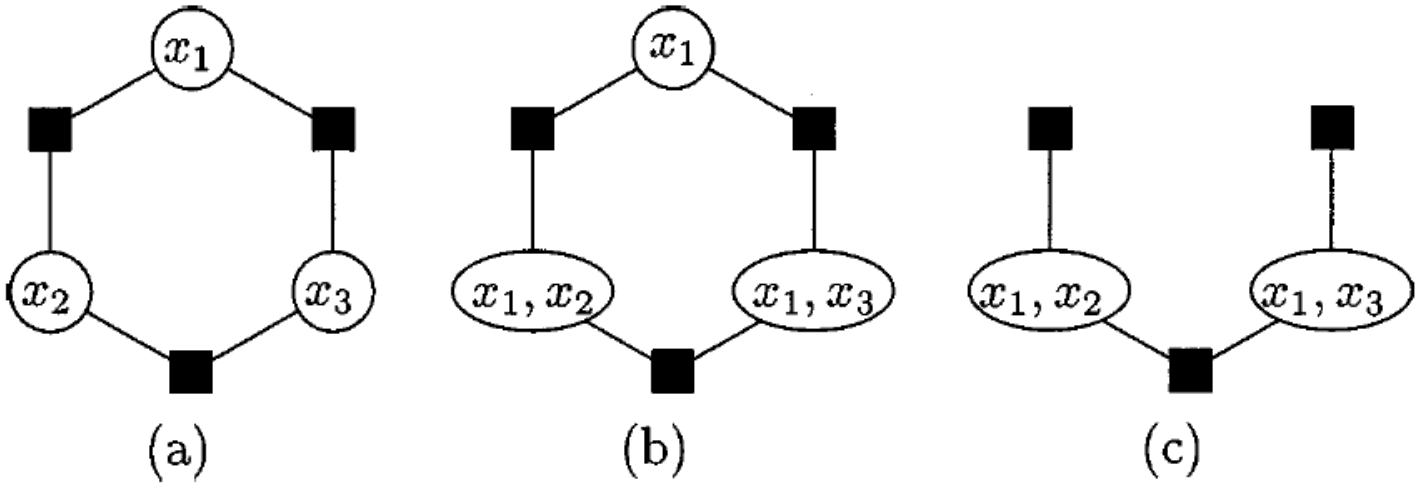
Transformations of Factor Graphs

- These transformations can be used to remove cycles
- Clustering



Transformations of Factor Graphs (cont'd)

- Stretching



Outline

Outline

Modeling Systems with FGs

- Definition: Iverson's convention. If P is a predicate (Boolean proposition), then

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

- Definition: characteristic function of a code with codebook C

$$\chi_C(c_1, c_2, \ldots, c_N) = [(c_1, c_2, \ldots, c_N) \in C]$$

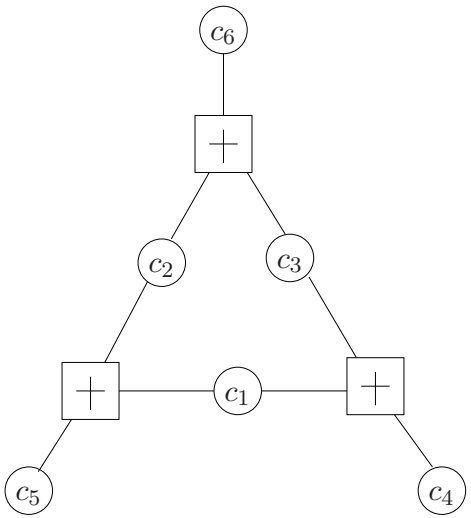
- Tanner graph for a linear code: it represents the characteristic function of the code. Ex.: a (6, 3) block code with parity check matrix $\mathbf{H}$ ($\mathbf{H}\mathbf{c}^T = \mathbf{0}$)

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Modeling Systems with FGs (cont'd)

- Membership in C is completely determined by checking whether each of the three equations is satisfied

$$\chi_C(c_1, c_2, c_3, c_4, c_5, c_6) = [(c_1, c_2, c_3, c_4, c_5, c_6) \in C]$$
$$[c_1 \oplus c_2 \oplus c_5 = 0][c_2 \oplus c_3 \oplus c_6 = 0][c_1 \oplus c_3 \oplus c_4 = 0]$$



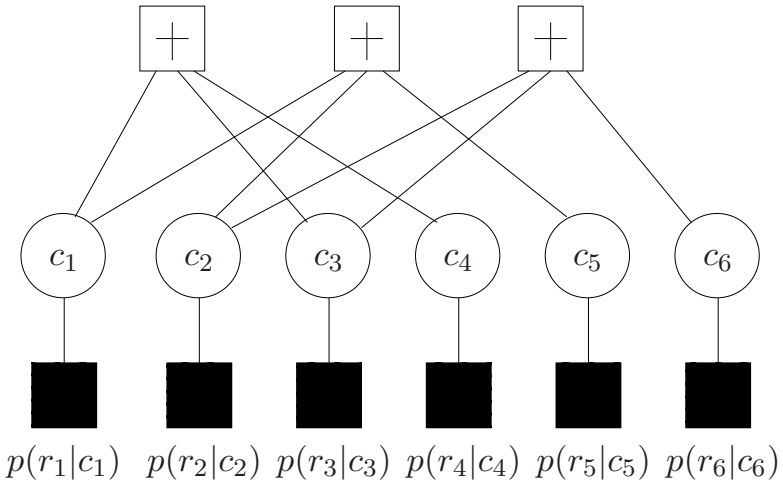
- Usually, this graph has cycles

Decoding of Block Codes (LDPC)

- In the MAP symbol decoding the aim is the computation of the marginal probabilities $P(c_n|\mathbf{r})$ from the joint pmf $P(\mathbf{c}|\mathbf{r})$
- On an AWGN channel with received samples $r_n = c_n + w_n$ $(n = 1, \ldots, N)$, we have

$$P(\mathbf{c}|\mathbf{r}) \propto P(\mathbf{c})p(\mathbf{r}|\mathbf{c}) = \frac{1}{|C|}\chi_C(\mathbf{c})\prod_{n=1}^{N} p(r_n|c_n)$$

where $p(r_n|c_n) = \frac{1}{2\pi\sigma^2}\exp\{-\frac{|r_n - c_n|^2}{2\sigma^2}\}$



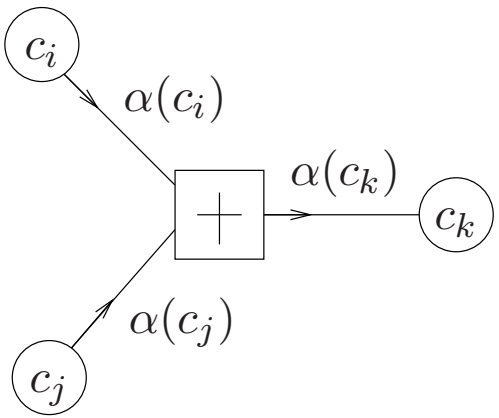
Decoding of Block Codes (LDPC) (cont'd)

- The application of the SP algorithm to this factor graph allows the computation of the marginal probabilities $P(c_n|\mathbf{r})$ (the code is assumed to be
- The graph has cycles: the algorithm proceeds iteratively until all checks are satisfied (or a maximum number of iterations is reached)
- Usually, the flooding schedule is adopted
- The messages on the edges of the graph are functions of discrete variables
- In the case of binary block codes, if log-likelihood ratios (LLR) are used instead of probabilities, the passed messages become real numbers

Simplifications for Binary Block Codes

- A variable node makes the product of the incoming messages) in terms of the LLR a variable node simply adds the incoming messages
- Check node to variable node

$$\begin{aligned}\alpha(c_k) &= \sum_{\sim\{c_k\}} \alpha(c_i)\alpha(c_j)[c_i \oplus c_j \oplus c_k = 0] \\ &= \sum_{c_i}\sum_{c_j} \alpha(c_i)\alpha(c_j)[c_i \oplus c_j \oplus c_k = 0]\end{aligned}$$



Simplifications for Binary Block Codes (cont'd)

- Hence

$$\begin{aligned}\alpha(c_k = 0) &= \alpha(c_i = 0)\alpha(c_j = 0) + \alpha(c_i = 1)\alpha(c_j = 1)\\ \alpha(c_k = 1) &= \alpha(c_i = 0)\alpha(c_j = 1) + \alpha(c_i = 1)\alpha(c_j = 0)\end{aligned}$$

- By defining

$$\ell_k = \ln \frac{\alpha(c_k = 0)}{\alpha(c_k = 1)}$$

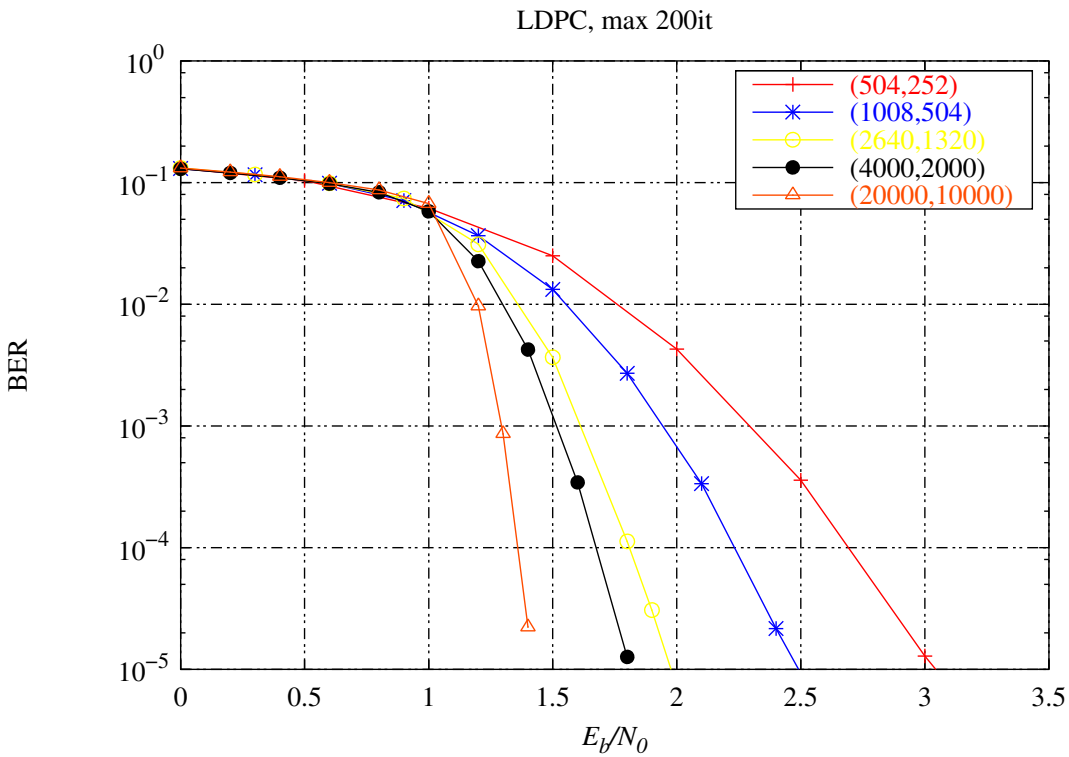
one has

$$\ell_k = 2\tanh^{-1}\left[\tanh\left(\frac{\ell_i}{2}\right)\tanh\left(\frac{\ell_j}{2}\right)\right] \simeq \operatorname{sgn}(\ell_i)\operatorname{sgn}(\ell_j)\min(|\ell_i|, |\ell_j|)$$

- This decoding algorithm is used to decode Low-Density Parity-Check (LDPC) codes, very long block codes with sparse (to reduce the decoding complexity) parity-check matrix
- In practice, only cycles of length 4 must be avoided

Performance of LDPC Codes

- Regular LDPC codes were proposed by Gallager in 1963



- Irregular LDPC codes, proposed in 2001, outperform the best known turbo codes

Outline

Decoding of Trellis Codes

- A generic code can be modeled as a finite state machine:

$$\begin{aligned} c_n &= g_1(a_n, \mu_n) \\ \mu_{n+1} &= g_2(a_n, \mu_n) \end{aligned}$$

- When this code is transmitted over an AWGN channel ($r_n = c_n + w_n$) we can write

$$\begin{aligned} P(\mathbf{c}, \mathbf{a}, \boldsymbol{\mu}|\mathbf{r}) &\propto p(\mathbf{r}|\mathbf{a}, \mathbf{c}, \boldsymbol{\mu}) P(\mathbf{c}, \boldsymbol{\mu}|\mathbf{a}) P(\mathbf{a}) \\ &= \Big[\prod_{n=1}^{N} p(r_n|c_n)\Big] P(\mathbf{c}, \boldsymbol{\mu}|\mathbf{a}) \Big[\prod_{n=1}^{N} P(a_n)\Big] \end{aligned}$$

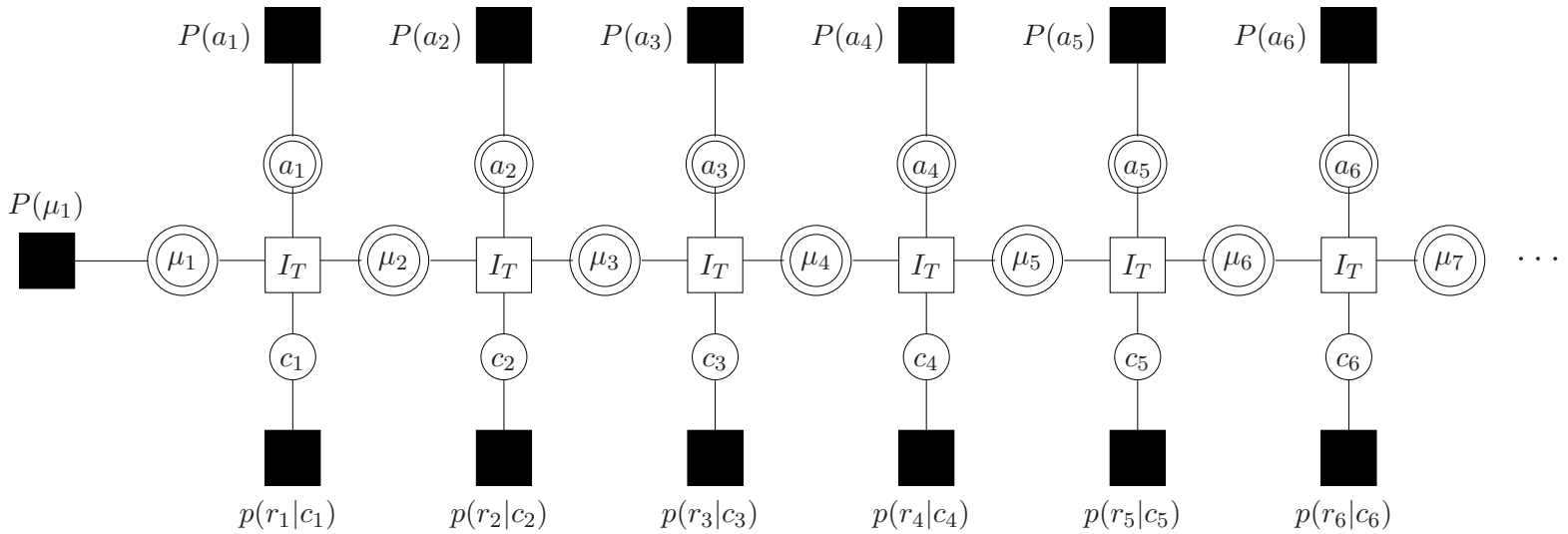
  - $P(\mathbf{c}, \boldsymbol{\mu}|\mathbf{a})$ is an indicator function, equal to one when $\mathbf{a}, \mathbf{c}$ and $\boldsymbol{\mu}$ are in a one-to-one correspondence, to zero otherwise. It can be expressed as the product of indicator functions for all trellis section (Wiberg graph)

$$P(\mathbf{c}, \boldsymbol{\mu}|\mathbf{a}) = P(\mu_1) \prod_{n=1}^{N} I_T(a_n, c_n, \mu_n, \mu_{n+1})$$

$$I_T(a_n, c_n, \mu_n, \mu_{n+1}) = [(a_n, c_n, \mu_n, \mu_{n+1}) \in T]$$

Decoding of Trellis Codes (cont'd)

$$P(\mathbf{c}, \mathbf{a}, \boldsymbol{\mu}|\mathbf{r}) \propto P(\mu_1) \prod_{n=1}^{N} p(r_n|c_n) I_T(a_n, c_n, \mu_n, \mu_{n+1}) P(a_n)$$



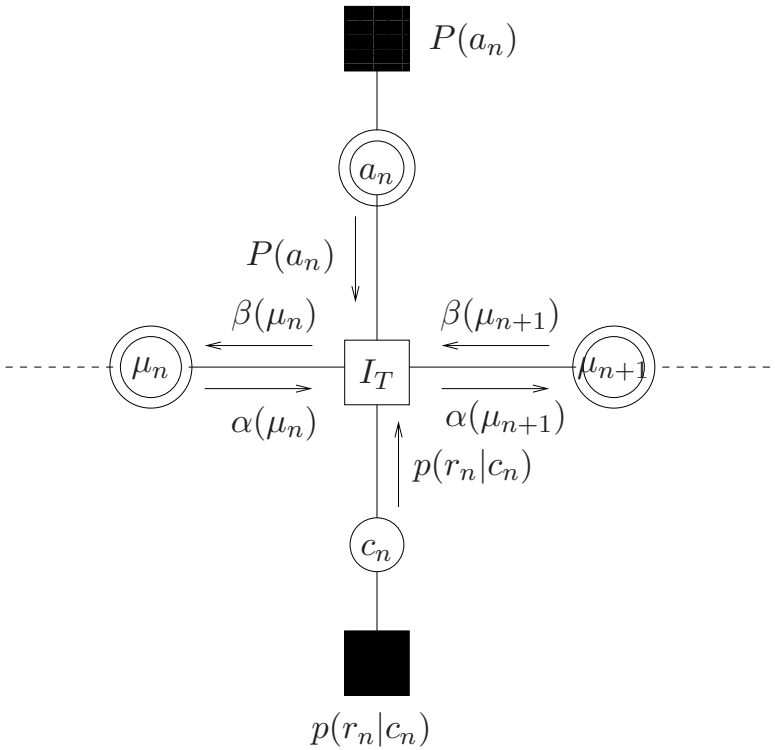
- The application of the SP algorithm leads to the BCJR algorithm
- The graph is cycle-free $\Rightarrow$ the exact marginal pmfs $P(a_n|\mathbf{r})$ are obtained (also pmfs $P(c_n|\mathbf{r})$ useful in serially concatenated schemes and $P(\mu_n|\mathbf{r})$ are also obtained)

Decoding of Trellis Codes (cont'd)

$$\alpha(\mu_{n+1}) = \sum_{\sim\{\mu_{n+1}\}} \alpha(\mu_n) p(r_n|c_n) I_T(a_n, c_n, \mu_n, \mu_{n+1}) P(a_n)$$

$$\beta(\mu_n) = \sum_{\sim\{\mu_n\}} \beta(\mu_{n+1}) p(r_n|c_n) I_T(a_n, c_n, \mu_n, \mu_{n+1}) P(a_n)$$

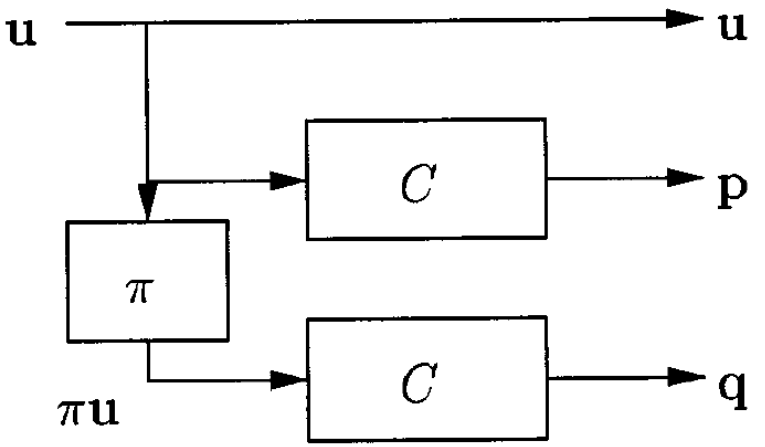
$$P(a_n|\mathbf{r}) = \sum_{\sim\{a_n\}} \alpha(\mu_n)\beta(\mu_{n+1}) p(r_n|c_n) I_T(a_n, c_n, \mu_n, \mu_{n+1}) P(a_n)$$

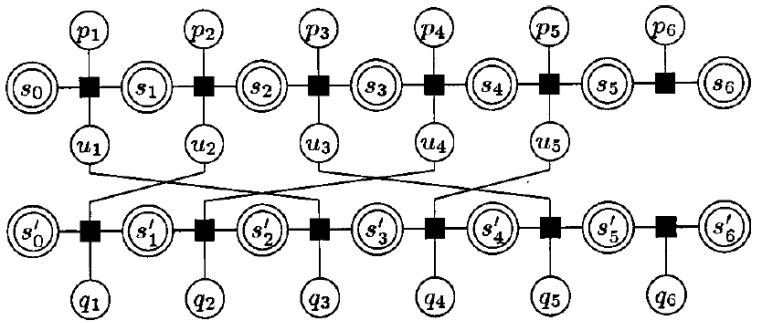


Outline

Iterative Decoding of Turbo Codes

- Decoding is not performed on the Wiberg graph of the overall code $\Rightarrow$ cycles $\Rightarrow$ iterative decoding





Outline

Detection over ISI Channels

- Received samples after the WMF: $r_n = \sum_{\ell=0}^{L} f_\ell a_{n-\ell} + w_n$
- Defining $\sigma_n = (a_{n-1}, a_{n-2}, \ldots, a_{n-L})$

$$\begin{aligned} P(\mathbf{a}, \boldsymbol{\sigma}|\mathbf{r}) &\propto p(\mathbf{r}|\mathbf{a}, \boldsymbol{\sigma}) P(\boldsymbol{\sigma}|\mathbf{a}) P(\mathbf{a}) \\ &= \Big[\prod_{n=1}^{N} p(r_n|a_n, \sigma_n)\Big] P(\boldsymbol{\sigma}|\mathbf{a}) \Big[\prod_{n=1}^{N} P(a_n)\Big] \end{aligned}$$

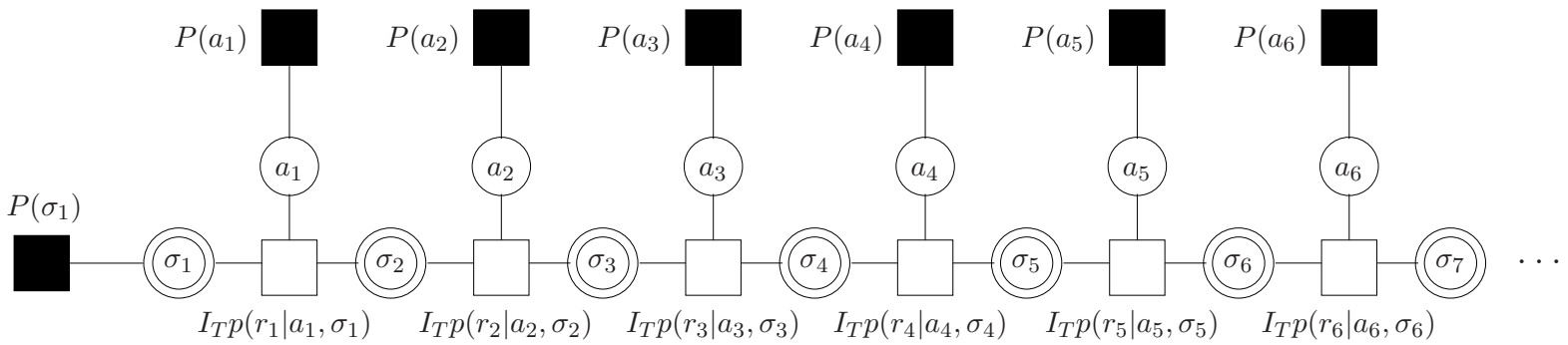
  - $p(r_n|a_n, \sigma_n) = \frac{1}{2\pi\sigma^2} \exp\{-|r_n - \sum_{\ell_0}^{L} f_\ell a_{n-\ell}|^2 / 2\sigma^2\}$
  - $P(\boldsymbol{\sigma}|\mathbf{a})$ is an indicator function, equal to one when $\mathbf{a}$ and $\boldsymbol{\sigma}$ are in a one-to-one correspondence, to zero otherwise:

$$P(\boldsymbol{\sigma}|\mathbf{a}) = P(\sigma_1) \prod_{n=1}^{N} I_T(a_n, \sigma_n, \sigma_{n+1})$$

$$I_T(a_n, \sigma_n, \sigma_{n+1}) = [(a_n, \sigma_n, \sigma_{n+1}) \in T]$$

Detection over ISI Channels (cont'd)

$$P(\mathbf{a}, \boldsymbol{\sigma}|\mathbf{r}) \propto P(\sigma_1) \prod_{n=1}^{N} p(r_n|a_n, \sigma_n) I_T(a_n, \sigma_n, \sigma_{n+1}) P(a_n)$$



- The application of the SP algorithm to this cycle-free FG still leads to the BCJR algorithm

Outline

Viterbi Algorithm

- This framework is based on the application of the distributive law
- The codomain of g can be any semiring with two operations "+" and "$\cdot$" that satisfy the distributive law
- We can use the "min-sum" semiring:

$$\begin{aligned} + &\rightarrow \min \\ \cdot &\rightarrow + \end{aligned}$$

- MAP sequence detection strategy can be expressed in the following equivalent ways

$$\begin{aligned} \hat{\mathbf{a}} &= \operatorname*{argmax}_{\mathbf{a}} P(\mathbf{a}|\mathbf{r}) = \operatorname*{argmin}_{\mathbf{a}} \Big[-\log P(\mathbf{a}|\mathbf{r}) \Big] \\ (\hat{\mathbf{a}}, \hat{\sigma}) &= \arg\min_{(\mathbf{a},\boldsymbol{\sigma})} \Big[-\log P(\mathbf{a}, \sigma|\mathbf{r}) \Big] \end{aligned}$$

since, given σ_1, $\mathbf{a}$ and $\boldsymbol{\sigma}$ are in a one-to-one correspondence

Viterbi Algorithm (cont'd)

- The problem

$$(\hat{x}_1, \hat{x}_2) = \arg \min_{(x_1, x_2)} f(x_1, x_2)$$

can be solved in two steps:

  - I step: compute

$$\hat{x}_2(x_1) = \operatorname{argmin}_{x_2} f(x_1, x_2)$$

  - II step: compute

$$\hat{x}_1 = \operatorname{argmin}_{x_1} f[x_1, \hat{x}_2(x_1)]$$

- Hence, in case of MAP sequence detection, decision on symbol a_k can be taken as

$$\hat{a}_k = \operatorname{argmin}_{a_k} \left\{ \min_{\sim\{a_k\}} \left[-\log P(\mathbf{a}, \sigma|\mathbf{r}) \right] \right\}$$

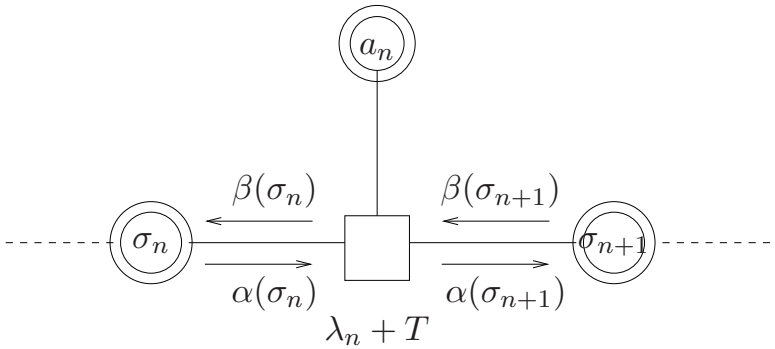
- $\min_{\sim\{a_k\}} \left[-\log P(\mathbf{a}, \sigma|\mathbf{r}) \right]$ is a marginalization in the "min-sum" semiring

Viterbi Algorithm (cont'd)

- Factors become addends

$$-\log P(\mathbf{a},\boldsymbol{\sigma}|\mathbf{r}) \propto \sum_{n=1}^{N} \Bigg[\underbrace{\frac{1}{2\sigma^2}\left| r_n - \sum_{\ell_0}^{L} f_\ell a_{n-\ell} \right|^2 - \log P(a_n)}_{\lambda_n(a_n,\sigma_n)} + T(a_n,\sigma_n,\sigma_{n+1}) \Bigg]$$

$$T(a_n,\sigma_n,\sigma_{n+1}) = -\log I_T(a_n,\sigma_n,\sigma_{n+1}) = \begin{cases} 0 & (a_n,\sigma_n,\sigma_{n+1}) \in T \\ +\infty & \text{otherwise} \end{cases}$$



Viterbi Algorithm (cont'd)

- Algorithm:

$$\begin{aligned}
\alpha(\sigma_{n+1}) &= \min_{a_n,\sigma_n}[\alpha(\sigma_n)+\lambda_n(a_n,\sigma_n)+T(a_n,\sigma_n,\sigma_{n+1})] \\
\beta(\sigma_n) &= \min_{a_n,\sigma_{n+1}}[\beta(\sigma_{n+1})+\lambda_n(a_n,\sigma_n)+T(a_n,\sigma_n,\sigma_{n+1})] \\
\hat{a}_n &= \operatorname*{argmin}_{a_n}\min_{\sigma_n,\sigma_{n+1}}[\alpha(\sigma_n)+\lambda_n(a_n,\sigma_n)+\beta(\sigma_{n+1})+T(a_n,\sigma_n,\sigma_{n+1})]
\end{aligned}$$

- The last equation is equivalent to compute the forward recursion up to $n = N$, to choose the best survivor, and then to come back taking decisions on it since $\forall n$

$$\min_{\sigma_{N+1}}\alpha(\sigma_{N+1}) = \min_{a_n,\sigma_n,\sigma_{n+1}}[\alpha(\sigma_n)+\lambda_n(a_n,\sigma_n)+\beta(\sigma_{n+1})+T(a_n,\sigma_n,\sigma_{n+1})]$$

Outline

Channels with Unknown Parameters

- We now consider an LDPC coded transmission over an unknown channel
- We will not consider non-Bayesian algorithms (using the “min-sum” semiring)
- We will consider Bayesian algorithms:
  - a priori average over channel parameters
  - numerical average over channel parameters (canonical distributions)
- The considered algorithms can employ soft estimations for code symbol probabilities available in iterative decoding schemes
- They can be used for turbo-codes also

System Model

- A sequence of M-ary coded symbols $\{c_n\}$, obtained from the encoding of a sequence of information bits and a proper mapping on a multilevel constellation, is transmitted from epoch 0 to epoch $N-1$
- To avoid phase ambiguity problems, pilot symbols or differential encoding may be also inserted in the sequence $\{c_n\}$
- A sequence of coded symbols is denoted in vector notation as

$$\mathbf{c}_{n_1}^{n_2} = (c_{n_1}, c_{n_1+1}, \ldots, c_{n_2}), \; n_2 > n_1$$

- The entire sequence is denoted by $\mathbf{c} = \mathbf{c}_0^{N-1}$
- This sequence is then modulated and transmitted over a channel which is modeled as a noiseless filter (possibly stochastic) plus additive white Gaussian noise (AWGN) with two-sided power spectral density $N_0/2$

System Model (cont'd)

- We assume that a single sample r_n is used for each coded symbol, which is practically sufficient in many cases
- We also assume that the channel is causal, that is $\mathbf{r}_0^n$ up to epoch n depends on the coded sequence up to epoch n only:

$$p(\mathbf{r}_0^n|\mathbf{c}) = p(\mathbf{r}_0^n|\mathbf{c}_0^n)$$

- This condition characterizes a noncoherent channel, a flat or a frequency-selective fading channel, and a channel with known and time-invariant ISI

System Model (cont'd)

- In the numerical results, we will focus on linear modulations on the:
  - noncoherent channel characterized by an unknown stochastic and possibly time-varying phase θ_n and model (true in the absence of strong phase variations):

    $$r_n = c_n e^{j\theta_n} + n_n$$

    For the phase noise process $\{\theta_n\}$, different statistics will be considered
  - flat fading channel

    $$r_n = h_n c_n + n_n$$

    where $\{h_n\}$ is a sequence of zero-mean complex Gaussian random variables with autocorrelation sequence $E\{h_n h_{n-k}^*\} = J_0(2\pi f_D T k)$, where $f_D T$ is the normalized Doppler rate

Factor Graph

- FG representing the joint APP of transmitted symbols $\{c_n\}$

$$P(\mathbf{c}|\mathbf{r}) \propto P(\mathbf{c})p(\mathbf{r}|\mathbf{c}) \propto \chi(\mathbf{c}) \prod_{n=0}^{N-1} p(r_n|\mathbf{r}_0^{n-1}, \mathbf{c}_0^n)$$

where $\chi(\mathbf{c})$ is the code indicator function, and the causality condition has been used

- Channels with finite memory (e.g., a channel with finite known ISI)

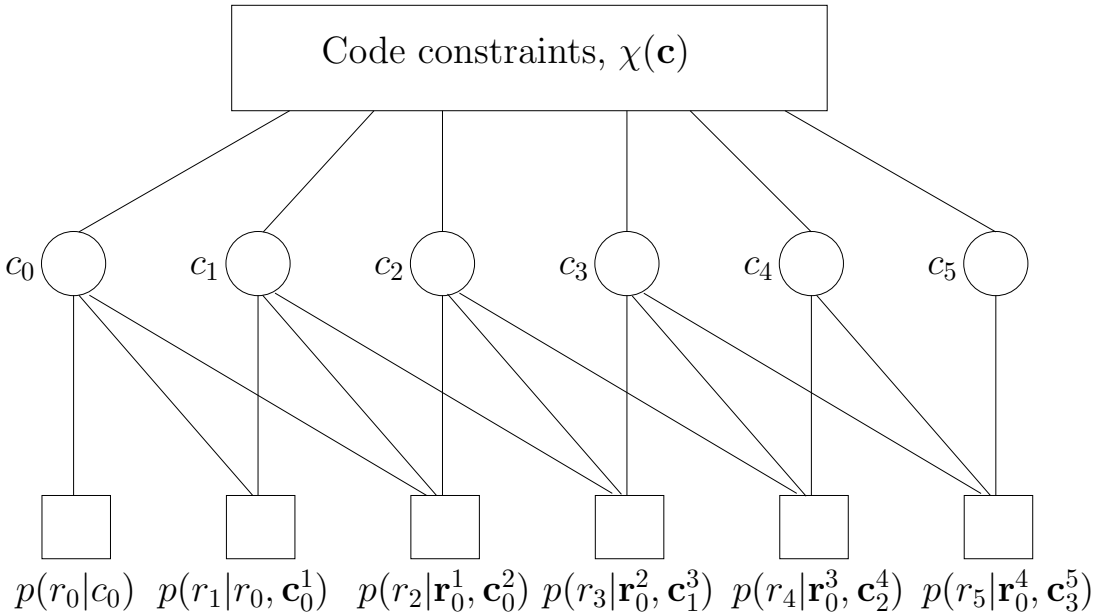
$$p(r_n|\mathbf{r}_0^{n-1}, \mathbf{c}_0^n) = p(r_n|\mathbf{r}_0^{n-1}, \mathbf{c}_{n-C}^n)$$

C is the finite memory parameter

$$P(\mathbf{c}|\mathbf{r}) \propto P(\mathbf{c})p(\mathbf{r}|\mathbf{c}) \propto \chi(\mathbf{c}) \prod_{n=0}^{N-1} p(r_n|\mathbf{r}_0^{n-1}, \mathbf{c}_{n-C}^n)$$

- The application of the SP algorithm to this FG allows us to compute the marginal APP $P(c_n|\mathbf{r}) \Rightarrow$ MAP symbol detection is implemented

Factor Graph (cont'd)



- Graph for $C = 2$
- With respect to a memoryless channel, we now have additional factor nodes which perform a marginalization, based on the channel model, without taking into account the code constraints
- The complexity of this marginalization is, in general, exponential in C

Factor Graph (cont'd)

- The finite memory condition is not verified in an exact sense for a noncoherent or a fading channel (channels with infinite memory)
- A FG may be devised but the complexity of the message computation at the factor nodes modeling the channel grows exponentially with n and thus becomes impractical
- An approximation is introduced assuming that r_n depends on the R most recent observations and the most recent $C \geq R$ coded symbols only
- This finite dependence property may be expressed as

$$p(r_n|\mathbf{r}_0^{n-1}, \mathbf{c}_0^n) \simeq p(r_n|\mathbf{r}_{n-R}^{n-1}, \mathbf{c}_{n-C}^n)$$

- This property, in general adopted in all practical detection schemes, is intuitive in the case of time-varying channels. In fact, in this case the conditional observations are asymptotically independent for increasing index difference

SP Algorithm

- The quality of the convergence of the SP algorithm to the exact marginal probabilities is in general determined by the girth of the graph
- As an example, in designing LDPC codes, cycles of length 4 must be avoided to ensure decoding convergence
- The graph derived from the proposed factorization has, in general, girth 4. However, we verified by computer simulations that these length-4 cycles involving two factor nodes which model the channel behavior often do not affect the convergence of the algorithm
- If this is not the case, as for transmissions over ISI channels, factor graph transformations can be adopted

SP ALGORITHM (cont'd)

- For the SP algorithm working on the described factor graphs, the most demanding computation is that performed at factor nodes modeling the channel. In fact, the marginalization performed by these nodes has in general a complexity which increases exponentially with C
- This complexity may be reduced by the following technique: by choosing an integer $Q < C$, we may compute the marginalization at factor nodes on the Q symbols with lowest reliabilities while the $C - Q$ symbols with highest reliabilities are hard-quantized on the basis of the messages on the graph
- The complexity becomes exponential in Q

Noncoherent Channels

- We model the channel phase as a time-invariant random variable with uniform distribution in $[0, 2\pi)$
- The finite dependence property will lead to a detection algorithm that can be used for time-varying channels also
- In this case $R = C$ and

$$p(r_n|\mathbf{r}_{n-C}^{n-1}, \mathbf{c}_{n-C}^{n}) = \frac{E_{\boldsymbol{\theta}_0^n}\{p(\mathbf{r}_{n-C}^{n}|\mathbf{c}_{n-C}^{n}, \boldsymbol{\theta}_0^n)\}}{E_{\boldsymbol{\theta}_0^{n-1}}\{p(\mathbf{r}_{n-C}^{n-1}|\mathbf{c}_{n-C}^{n-1}, \boldsymbol{\theta}_0^{n-1})\}}$$

$$= \frac{\mathrm{I}_0\left(\frac{1}{\sigma^2}\left|\sum_{i=0}^{C} r_{n-i}c_{n-i}^*\right|\right)}{\mathrm{I}_0\left(\frac{1}{\sigma^2}\left|\sum_{i=1}^{C} r_{n-i}c_{n-i}^*\right|\right)} e^{-\frac{|c_n|^2}{2\sigma^2}}$$

Time-Varying Noncoherent Channels

- In the case of the Wiener model ($\theta_n = \theta_{n-1} + \Delta_n$), an exact closed form expression of $E_{\boldsymbol{\theta}_0^n}\{p(\mathbf{r}_{n-C}^n|\mathbf{c}_{n-C}^n, \boldsymbol{\theta}_0^n)\}$ and $E_{\boldsymbol{\theta}_0^{n-1}}\{p(\mathbf{r}_{n-C}^{n-1}|\mathbf{c}_{n-C}^{n-1}, \boldsymbol{\theta}_0^{n-1})\}$, does not exist. However, a very good approximation can be found. By using an approximate result on Tikhonov distributions, it is possible to express

$$E_{\boldsymbol{\theta}_0^n}\{p(\mathbf{r}_{n-C}^n|\mathbf{c}_{n-C}^n, \boldsymbol{\theta}_0^n)\} \simeq \prod_{i=n-C}^{n} \mathrm{I}_0(|z_i|) e^{-\frac{|c_i|^2}{2\sigma^2}} \prod_{i=n-C+1}^{n} \frac{1}{\mathrm{I}_0\left(\frac{|z_i|}{1+\sigma_\Delta^2|z_i|}\right)}$$

where coefficients z_i can be recursively computed as

$$\begin{aligned} z_n &= \frac{r_n c_n^*}{\sigma^2} \\ z_{i-1} &= \frac{z_i}{1+\sigma_\Delta^2|z_i|} + \frac{r_{i-1}c_{i-1}^*}{\sigma^2}, \\ & \qquad i = n, n-1, \ldots, n-C+1 . \end{aligned}$$

Time-Varying Channels (cont'd)

- For a time-varying phase process θ_n, assumed stationary, zero-mean and described by a given autocorrelation sequence of the phasor process $e^{j\theta_n}$, an approximate linear predictive approach may be adopted which allows to increase the receiver robustness
- In this case, omitting irrelevant constant terms,

$$p(r_n|\mathbf{r}_{n-C}^{n-1}, \mathbf{c}_{n-C}^{n}) \simeq \exp\left\{ -\frac{1}{\sigma_e^2} \left| r_n - c_n \frac{\sum_{i=1}^{C} p_i \frac{r_{n-i}}{c_{n-i}}}{|\sum_{i=1}^{C} p_i \frac{r_{n-i}}{c_{n-i}}|} \right|^2 \right\}$$

where the prediction coefficients $\{p_i\}_{i=1}^{C}$ and the mean square prediction error σ_e^2 are obtained by solving a Wiener-Hopf linear system

Time-Varying Channels (cont'd)

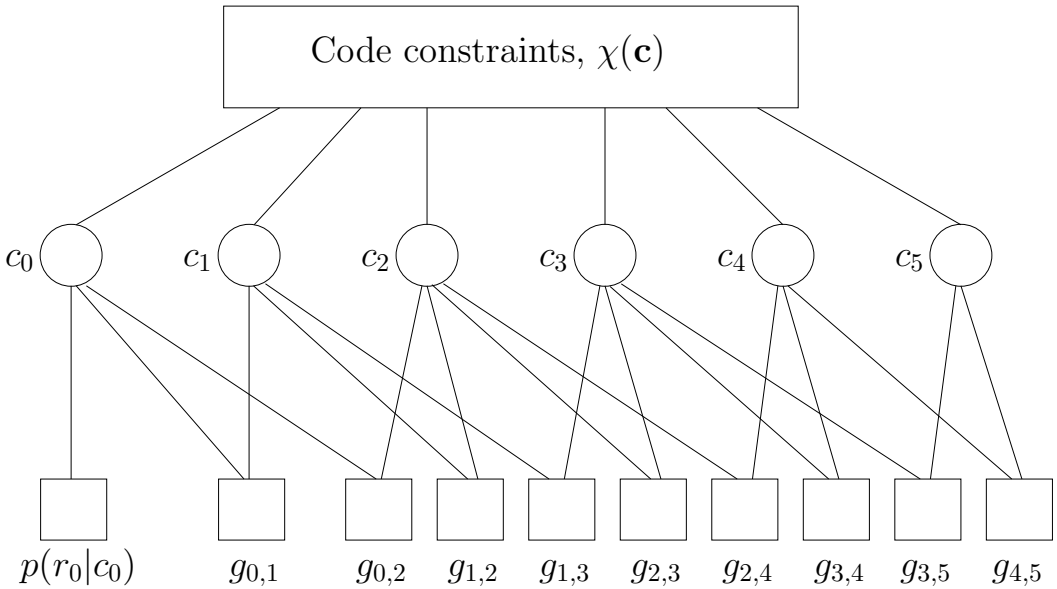
- For PSK signals, the prediction coefficients become independent of the considered sequence
- In addition, approximating $|\sum_{i=1}^{C} p_i \frac{r_{n-i}}{c_{n-i}}| \simeq |\sum_{i=1}^{C} p_i|$, and taking into account that $|c_n| = 1$, we may express

$$p(r_n|\mathbf{r}_{n-C}^{n-1}, \mathbf{c}_{n-C}^{n}) \propto \prod_{i=1}^{C} \exp\left\{ \frac{2}{\sigma_e^2 |\sum_{i=1}^{C} p_i|} \mathrm{Re}\left[p_i r_n c_n^* r_{n-i}^* c_{n-i}\right] \right\}$$

$$= \prod_{i=1}^{C} g_{i,n}(c_{n-i}, c_n)$$

- This further factorization as a direct impact on the graph structure

Time-Varying Channels (cont'd)



- Each factor node can be decomposed in C simpler degree-2 factor nodes
- Hence, for increasing values of C, the number of factor nodes increases linearly but the computational burden at each factor node remains the same
- Note that in this modified factor graph there are no cycles of length 4

Flat Fading Channels

- For a flat fading channel, based on linear prediction we have

$$p(r_n|\mathbf{r}_{n-C}^{n-1}, \mathbf{c}_{n-C}^{n}) \propto \exp\left\{-\frac{1}{\sigma_e^2}\left|r_n - c_n \sum_{i=1}^{C} p_i \frac{r_{n-i}}{c_{n-i}}\right|^2\right\}.$$

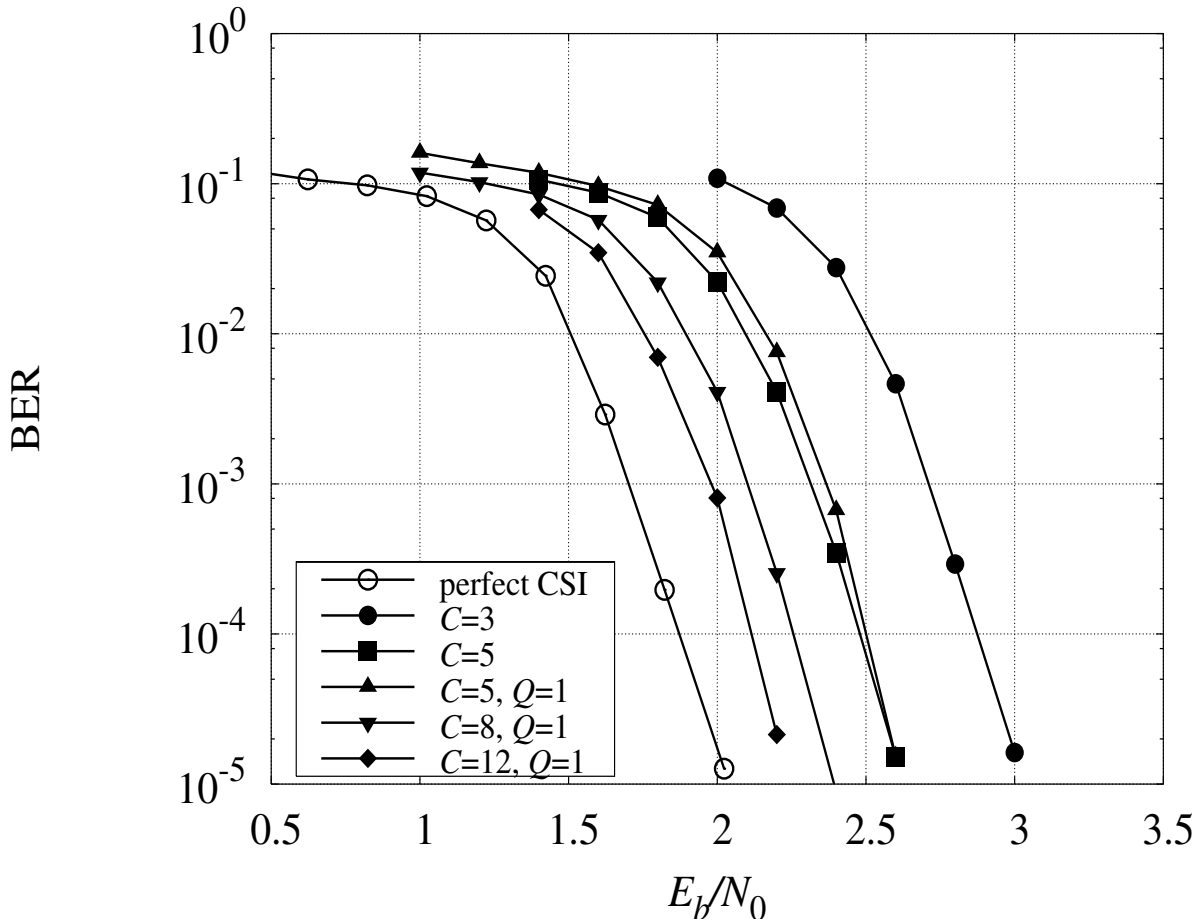
- For PSK signals, the prediction coefficients become independent of the considered sequence. After straightforward manipulations we have

$$P(\mathbf{c}|\mathbf{r}) \simeq \chi(\mathbf{c}) \prod_{n=0}^{N-1} \prod_{i=1}^{C} \exp\left\{\frac{2}{\sigma_e^2}\mathrm{Re}[q_i r_n r_{n-i}^* c_n^* c_{n-i}]\right\}$$

  where $q_i = p_i - \sum_{\ell=1}^{C-i} p_\ell p_{\ell+i}$.

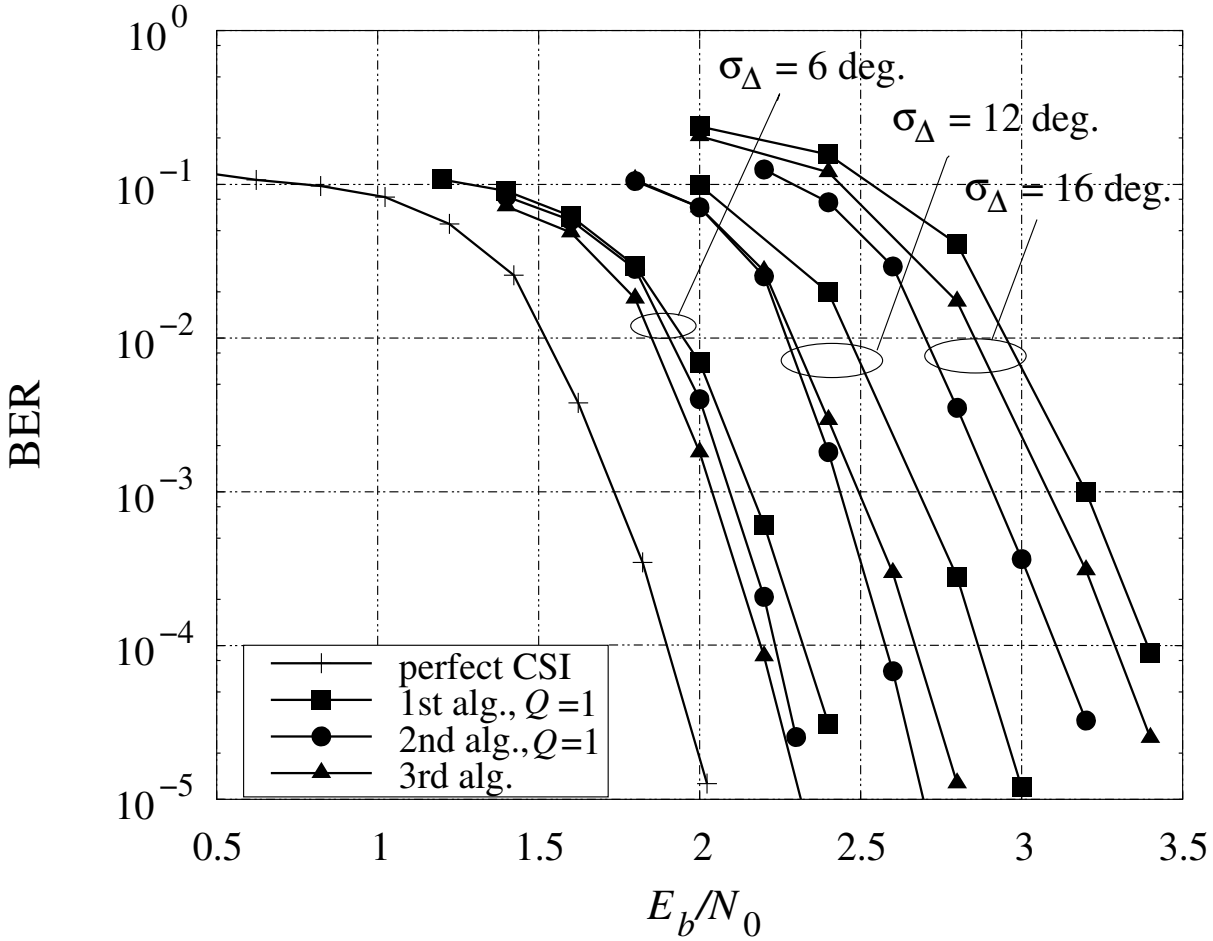
- In this case also, each factor node can be decomposed in C simpler degree-2 factor nodes. For increasing values of C, the number of factor nodes increases linearly but the computational burden at each factor node remains the same

Numerical Results (1)



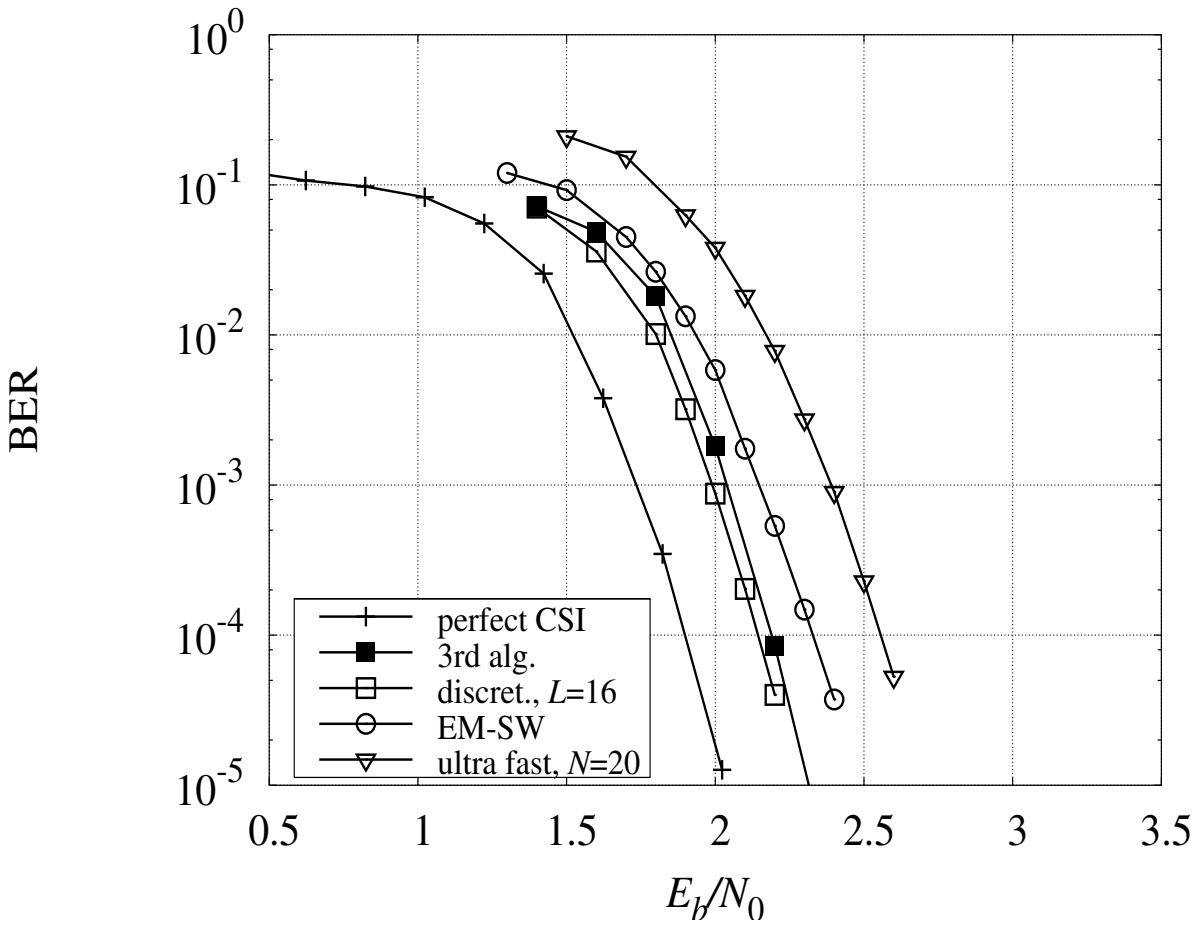
- Time-invariant phase
- (3,6)-regular LDPC code with codewords of length 4000
- BPSK modulation
- Maximum of 200 iterations of the SP algorithm on the overall graph
- Flooding schedule
- A pilot symbol every 19 code bits to avoid ambiguity $\Rightarrow$ increase in the required SNR of 0.223 dB

Numerical Results (2)



- Same code, modulation format, and number of pilots
- Time-varying Wiener phase

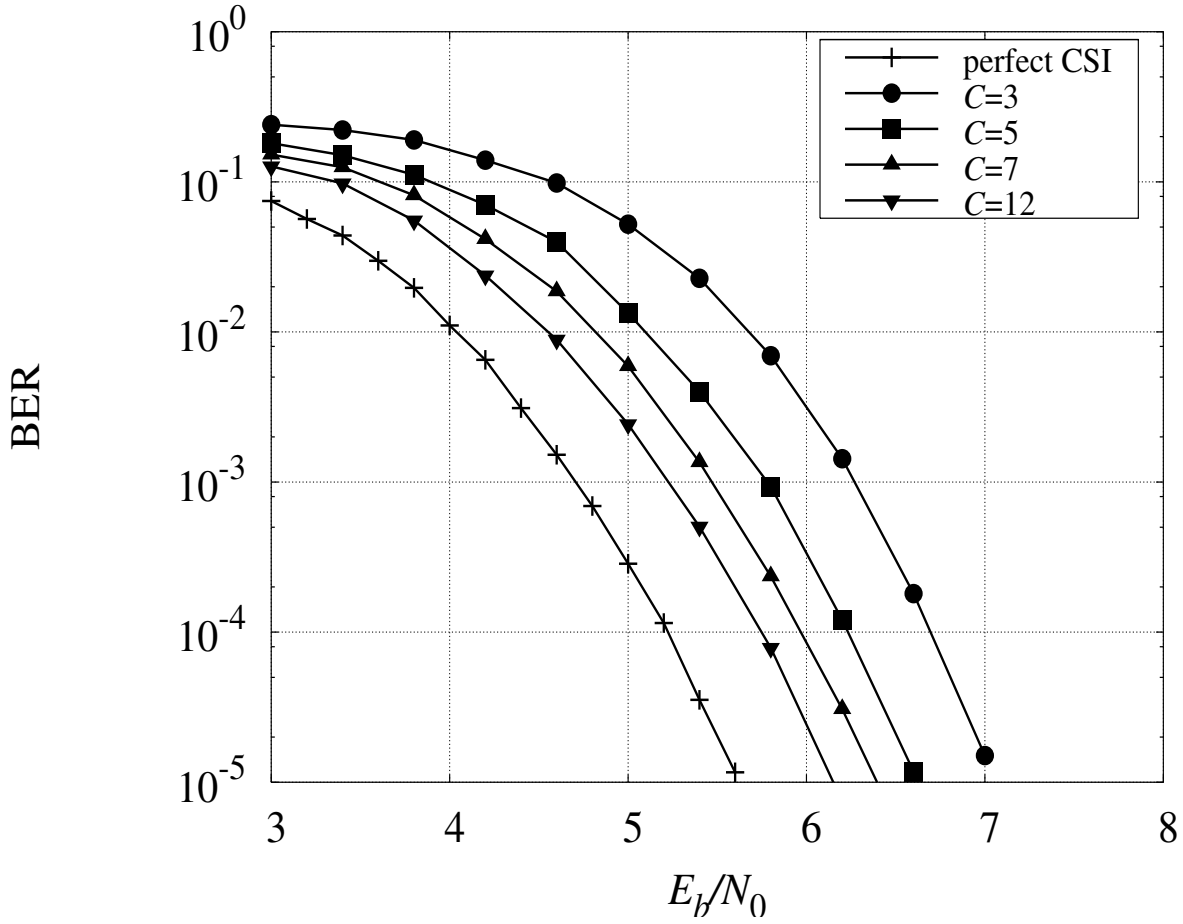
Numerical Results (3)



- Same code, modulation format, and number of pilots
- Time-varying Wiener phase with $\sigma_\Delta = 6$ deg.
- Comparison with other solutions:
  - Phase distretization
  - EM-SW
  - ultra-fast

  (Motedayen-Aval & Anastasopoulos)

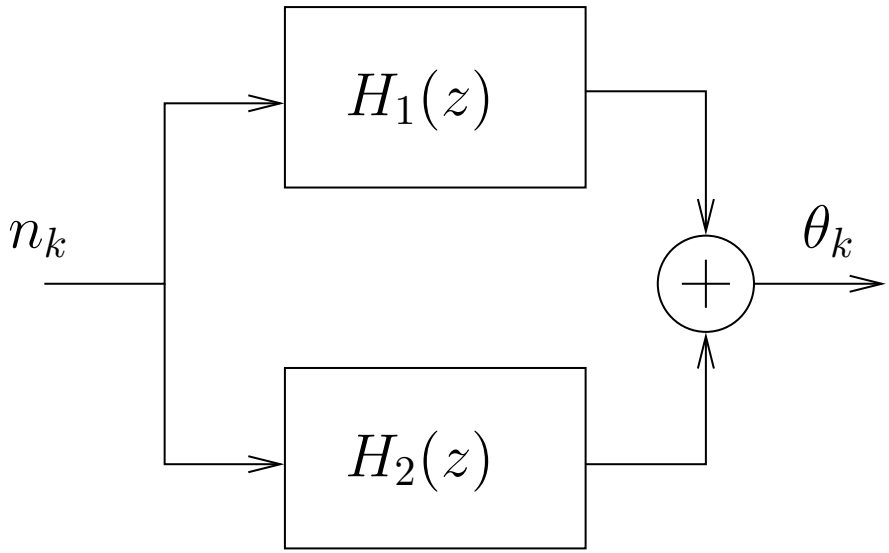
Numerical Results (4)



- Same code, modulation format, and number of pilots
- Flat fading channel with $f_D T = 10^{-}2$

System Model: Phase Noise

- Phase models:
  - Wiener model $\theta_k = \theta_{k-1} + \Delta_k$ where Δ_k i.i.d. Gaussian increments with zero mean and standard deviation σ_Δ
  - Constant phase as a particular case ($\sigma_\Delta = 0$)
  - ESA model (DVB-S2 compliant)



n_k discrete Gaussian white process, $H_1(z)$ and $H_2(z)$ IIR filters chosen to fit an experimental phase noise mask

Numerical Average over Channel Parameters

- Variable nodes representing the channel parameters are explicitly introduced in the FG by considering the joint distribution of symbols and unknown parameters and the corresponding FG
- For the computation of the marginal pmfs $P(c_k|\mathbf{r})$, the average over channel parameters of the joint conditional distribution of $\mathbf{c}$ and θ is now performed by the SP algorithm
- Approach to handle the presence in the graph of continuous rvs (representing the channel parameters):
  - Canonical distributions

Overall Factor Graph

- The joint a posteriori distribution of symbols and unknown parameters may be expressed as (Wiener model)

$$\begin{aligned} p(\mathbf{c},\boldsymbol{\theta}|\mathbf{r}) &\propto p(\mathbf{r}|\mathbf{c},\boldsymbol{\theta})\chi(\mathbf{c})p(\boldsymbol{\theta}) \\ &= \chi(\mathbf{c})p(\theta_0)\prod_{k=0}^{N-1} p(r_k|c_k,\theta_k)\prod_{k=1}^{N-1} p(\theta_k|\theta_{k-1}) \\ &\propto \chi(\mathbf{c})p(\theta_0)\prod_{k=0}^{N-1} f_k(c_k,\theta_k)\prod_{k=1}^{N-1} p(\theta_k|\theta_{k-1}) \end{aligned}$$

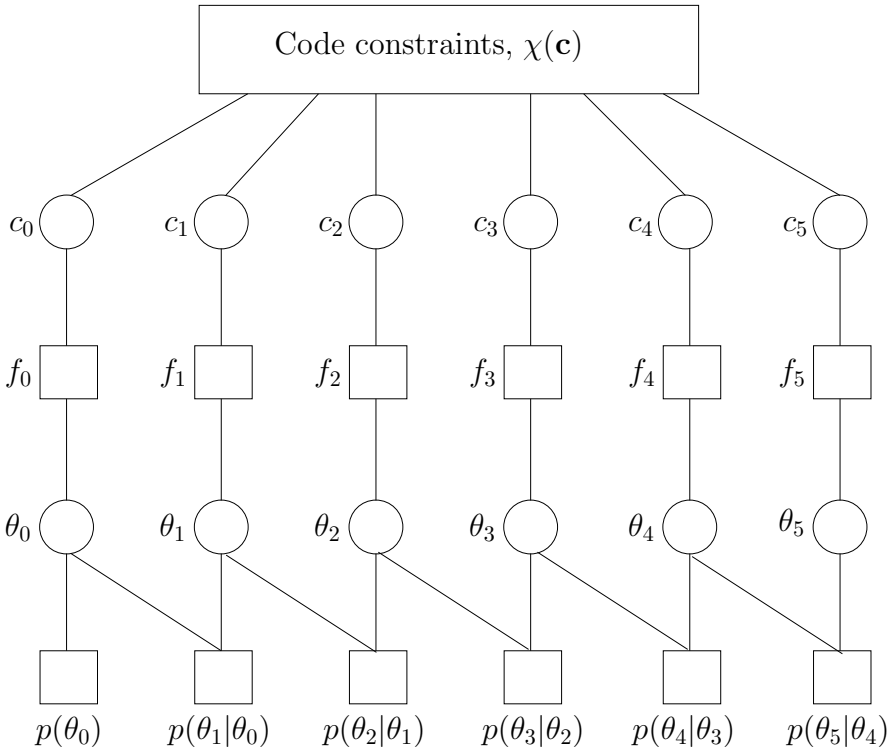
where

$$f_k(c_k,\theta_k) = \exp\left\{-\frac{1}{2\sigma^2}|r_k - c_k e^{j\theta_k}|^2\right\}$$

$$p(\theta_0) = \frac{1}{2\pi} \quad , \quad \theta_0 \in [0, 2\pi)$$

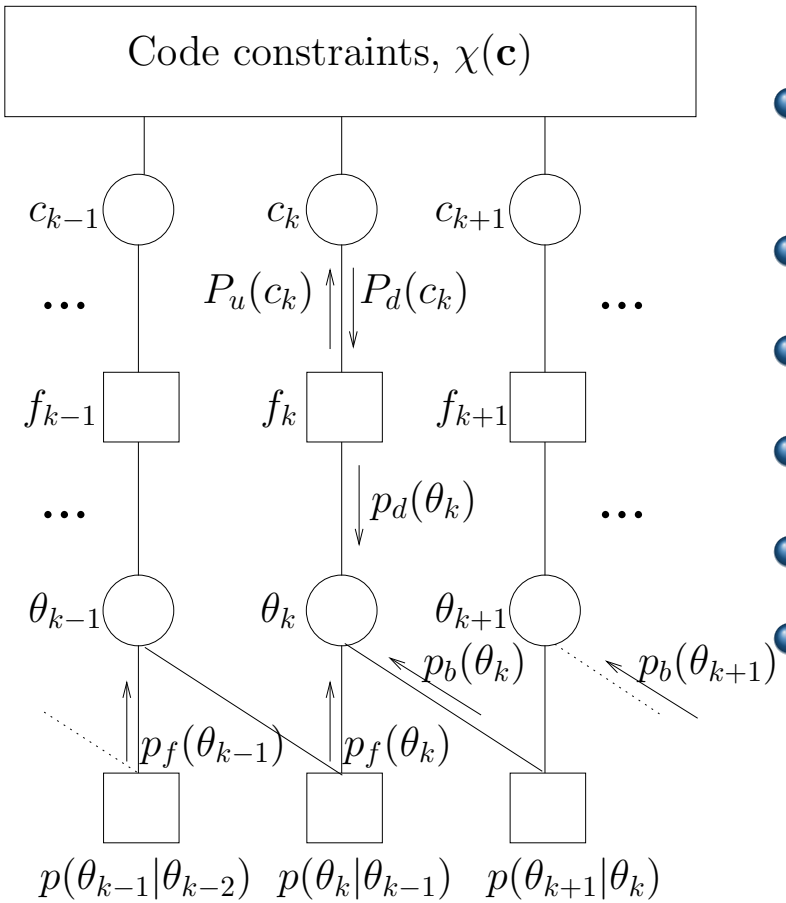
$$p(\theta_k|\theta_{k-1}) = p_\Delta(\theta_k - \theta_{k-1}) = \frac{1}{\sqrt{2\pi\sigma_\Delta^2}} e^{-\frac{(\theta_k-\theta_{k-1})^2}{2\sigma_\Delta^2}}$$

Overall Factor Graph (cont'd)



- Corresponding FG

Overall Factor Graph (cont'd)

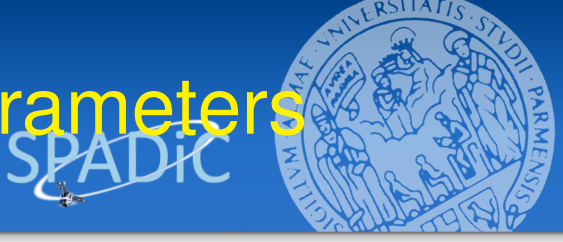


- Omitting the esplicit reference to the current iteration
- $p_d(\theta_k) = \sum_{c \in \mathcal{C}} P_d(c_k = c) f_k(c_k = c, \theta_k)$
- $p_f(\theta_k) = \int_0^{2\pi} p_d(\theta_{k-1}) p_f(\theta_{k-1}) p(\theta_k|\theta_{k-1})\, d\theta_{k-1}$
- $p_b(\theta_k) = \int_0^{2\pi} p_d(\theta_{k+1}) p_b(\theta_{k+1}) p(\theta_{k+1}|\theta_k)\, d\theta_{k+1}$
- $P_u(c_k) = \int_0^{2\pi} p_f(\theta_k) p_b(\theta_k) f_k(c_k, \theta_k)\, d\theta_k$
- The optimal SP algorithm is unfeasible

Canonical Distributions

- We represent the pdfs, which are the messages sent or received from nodes representing the channel parameters, with given canonical pdfs, described by some parameters
- This representation can be exact or, more often, involve some approximate assumptions
- As an example, we could assume that these messages are Gaussian pdfs which can be completely specified by their mean and variance
- Hence, the SP algorithm has just to forward the parameters of the distribution
- Quantization-based algorithm, Fourier-based algorithm and Tikhonov algorithm

Quantization of the Channel Parameters



- If we quantize the channel parameters, we may consider the factor graph representing the joint APP $P(\mathbf{c}, \theta|\mathbf{r})$ (Worthen-Stark, IEEE Trans. Inf. Theory Feb. 2001)
- The application of the SP algorithm to this factor graph allows us to compute the desired marginal APPs $P(c_k|\mathbf{r})$ and the collateral ones $P(\theta_k|\mathbf{r})$
- From a point of view of the involved approximation, in this case we have the quantization of in general continuous channel parameters and their statistics
- This approach becomes optimal (in the sense that it approaches the performance of the exact SP algorithm) for a sufficiently large number of quantization levels, at the expenses of an increased computational complexity
- As an example, for M-PSK signals, $L = 8M$ quantization levels are sufficient

Quantization (cont'd)

- In the case of the Wiener model

$$\begin{aligned} P(\mathbf{c},\boldsymbol{\theta}|\mathbf{r}) &\propto \chi(\mathbf{c})P(\boldsymbol{\theta})p(\mathbf{r}|\mathbf{c},\boldsymbol{\theta}) \\ &\propto \chi(\mathbf{c})P(\theta_0)\prod_{k=1}^{N-1}P(\theta_k|\theta_{k-1})\prod_{k=0}^{N-1}f_k(c_k,\theta_k) \end{aligned}$$

  where

$$f_k(c_k,\theta_k) = \exp\left\{-\frac{1}{2\sigma^2}|r_k - c_k e^{j\theta_k}|^2\right\}$$

- The pmf $P(\theta_k|\theta_{k-1})$ is related to the quantized distribution of the increment Δ_k:

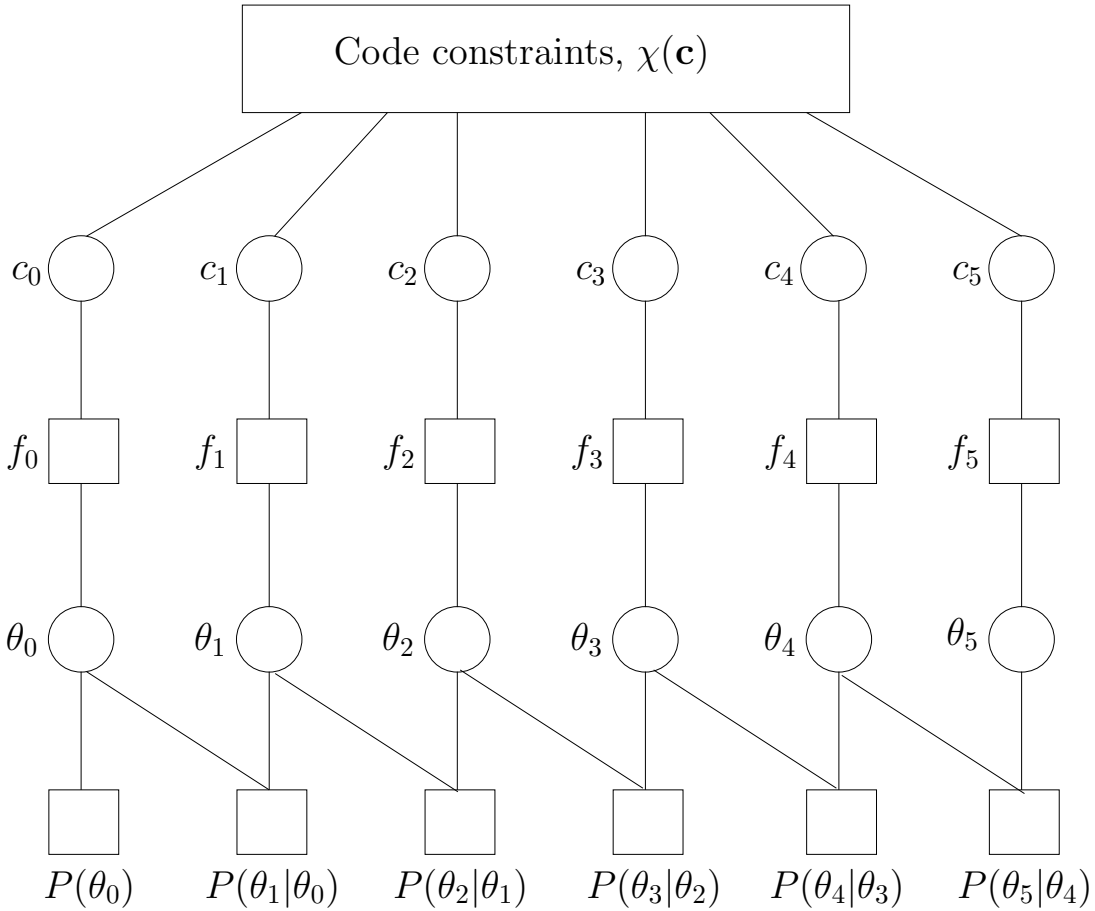
$$P(\theta_k|\theta_{k-1}) = P_\Delta(\theta_k - \theta_{k-1})$$

- As an example, for $\sigma_\Delta \ll 1$

$$P_\Delta(\Delta_k) = \begin{cases} 1-\sigma_\Delta^2\left(\frac{L}{2\pi}\right)^2 & \text{for } \Delta_k = 0 \\ \frac{\sigma_\Delta^2}{2}\left(\frac{L}{2\pi}\right)^2 & \text{for } \Delta_k = \pm\frac{2\pi}{L} \end{cases}$$

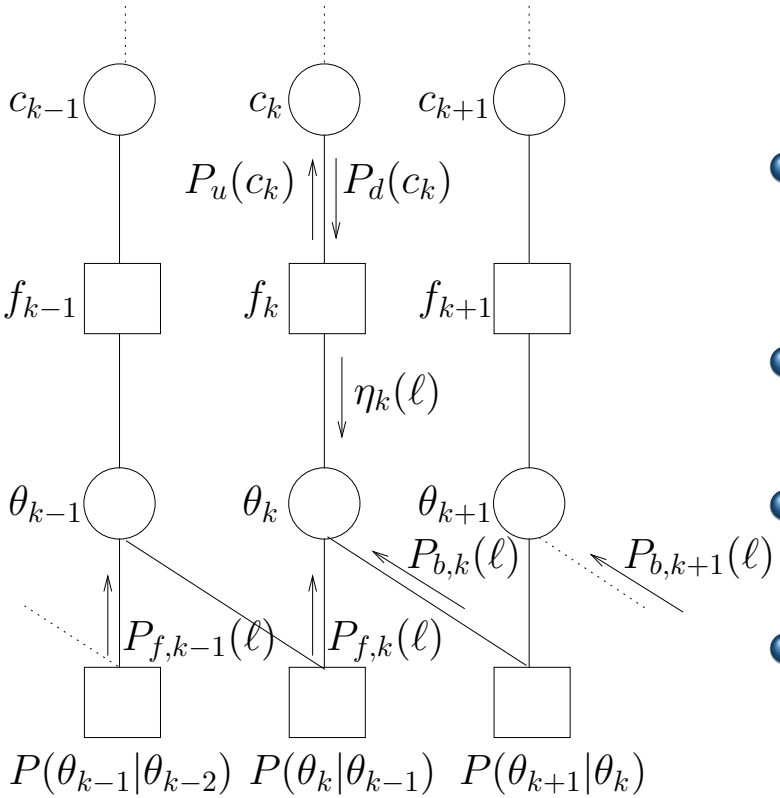
  (discrete random walk approximation)

Quantization (cont'd)



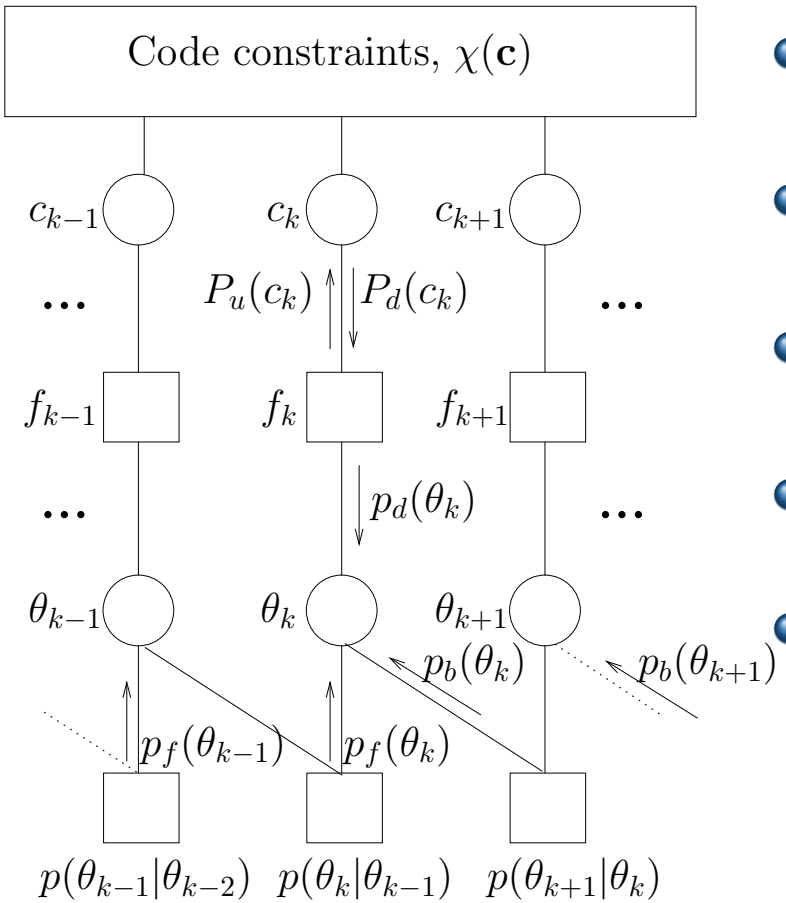
- Corresponding FG

Quantization (cont'd)



- Omitting the esplicit reference to the current iteration
- $\eta_k(\ell) = \sum_{c \in \mathcal{C}} f_k(c_k = c, 2\pi\ell/L) P_d(c_k = c)$
  $\ell = 0, 1, \ldots, L-1$
- $P_{f,k}(\ell) = \sum_{m=0}^{L-1} P_{f,k-1}(m) \eta_{k-1}(m) P_\Delta\left(2\pi \frac{\ell - m}{L}\right)$
- $P_{b,k}(\ell) = \sum_{m=0}^{L-1} P_{b,k+1}(m) \eta_{k+1}(m) P_\Delta\left(2\pi \frac{\ell - m}{L}\right)$
- $P_u(c_k) = \sum_{\ell=0}^{L-1} P_{f,k}(\ell) P_{b,k}(\ell) f_k(c_k, 2\pi\ell/L)$

Fourier-Based Algorithm



- $p_d(\theta_k) = \sum_{c \in \mathcal{C}} P_d(c_k = c) f_k(c_k = c, \theta_k)$
- $p_f(\theta_k) = \int_0^{2\pi} p_d(\theta_{k-1}) p_f(\theta_{k-1}) p(\theta_k|\theta_{k-1})\, d\theta_{k-1}$
- $p_b(\theta_k) = \int_0^{2\pi} p_d(\theta_{k+1}) p_b(\theta_{k+1}) p(\theta_{k+1}|\theta_k)\, d\theta_{k+1}$
- $P_u(c_k) = \int_0^{2\pi} p_f(\theta_k) p_b(\theta_k) f_k(c_k, \theta_k)\, d\theta_k$
- All these pdfs are periodic in θ_k with period 2π. Hence, they can be expanded in Fourier series

Fourier-Based Algorithm (cont'd)

- It is possible to show that $p_d(\theta_k)$ can be expressed as

$$p_d(\theta_k) = \sum_{\ell=-\infty}^{\infty} A_k^{(\ell)} e^{j\ell\theta_k}$$

having defined

$$A_k^{(\ell)} = \sum_{c\in\mathcal{C}} P_d(c_k = c) e^{-\frac{|c|^2}{2\sigma^2}} \mathrm{I}_\ell\left(\frac{|r_k||c|}{\sigma^2}\right) e^{-j\ell\phi(r_k c^*)}$$

where $\mathrm{I}_\ell(x)$ is the modified Bessel function of the first kind of order ℓ and for a complex number z, $\phi(z) = \arg(z)$

- For M-PSK signals, the expression of coefficients $A_k^{(\ell)}$, simplifies to

$$A_k^{(\ell)} = e^{-j\ell\phi(r_k)} \mathrm{I}_\ell\left(\frac{|r_k|}{\sigma^2}\right) \sum_{c\in\mathcal{C}} P_d(c_k = c) c^\ell$$

Fourier-Based Algorithm (cont'd)

- We define $B_{f,k}^{(\ell)}$ and $B_{b,k}^{(\ell)}$ as the Fourier coefficients of pdfs $p_f(\theta_k)$ and $p_b(\theta_k)$, i.e.,

$$p_f(\theta_k) = \sum_{\ell=-\infty}^{\infty} B_{f,k}^{(\ell)} e^{j\ell\theta_k} \quad , \quad p_b(\theta_k) = \sum_{\ell=-\infty}^{\infty} B_{b,k}^{(\ell)} e^{j\ell\theta_k}$$

- The integral recursive equation for $p_f(\theta_k)$ becomes

$$B_{f,k}^{(\ell)} = D_\ell(\sigma_\Delta) \sum_{m=-\infty}^{\infty} A_{k-1}^{(m)} B_{f,k-1}^{(\ell-m)} = D_\ell(\sigma_\Delta)[A_{k-1}^{(\ell)} \otimes B_{f,k-1}^{(\ell)}]$$

  where $\otimes$ denotes convolution between sequences and

$$D_\ell(\sigma_\Delta) = e^{-\frac{\sigma_\Delta^2 \ell^2}{2}}$$

- The starting condition is $B_{f,0}^{(\ell)} = \delta(\ell)$, where $\delta(\ell)$ denotes the Kronecker delta
- Similarly, to compute coefficients $\{B_{b,k}^{(\ell)}\}$, we have the following backward recursion:

$$B_{b,k}^{(\ell)} = D_\ell(\sigma_\Delta)[A_{k+1}^{(\ell)} \otimes B_{b,k+1}^{(\ell)}]$$

Fourier-Based Algorithm (cont'd)

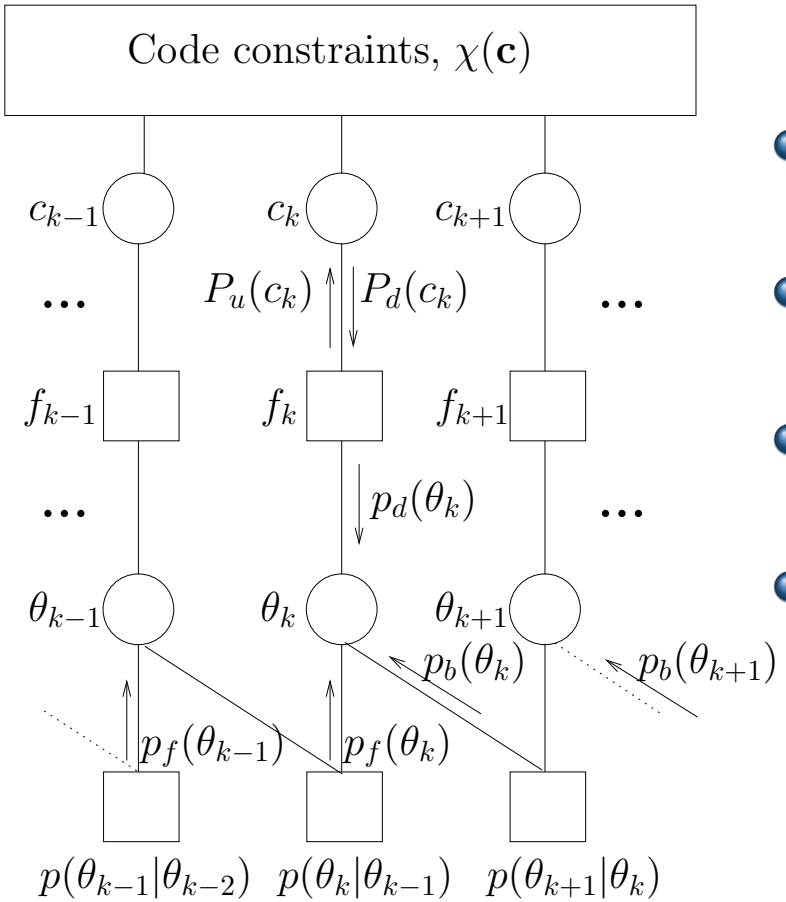
- Finally

$$P_u(c_k) = e^{-\frac{|c_k|^2}{2\sigma^2}} \left\{ \sum_{m=-\infty}^{\infty} B_{f,k}^{(m)} \sum_{n=-\infty}^{\infty} B_{b,k}^{(n-m)} \mathrm{I}_n \left(\frac{|r_k||c_k|}{\sigma^2} \right) e^{jn\phi(r_k c_k^*)} \right\}$$

- In general, a reduced number N_c of coefficients must be taken into account
- For a binary modulation format, this algorithm has practically the same complexity of the quantization-based algorithm
- For a modulation format characterized by a more dense constellation, if for the quantization-based algorithm the optimal number of quantization levels, and thus the complexity, must be increased, the number N of considered Fourier coefficients in this new algorithm remains practically the same

Tikhonov Algorithm

- Is it possible to substantially reduce the complexity?



- $p_d(\theta_k) = \sum_{c \in \mathcal{C}} P_d(c_k = c) f_k(c_k = c, \theta_k)$
- $p_f(\theta_k) = \int_0^{2\pi} p_d(\theta_{k-1}) p_f(\theta_{k-1}) p(\theta_k|\theta_{k-1})\, d\theta_{k-1}$
- $p_b(\theta_k) = \int_0^{2\pi} p_d(\theta_{k+1}) p_b(\theta_{k+1}) p(\theta_{k+1}|\theta_k)\, d\theta_{k+1}$
- $P_u(c_k) = \int_0^{2\pi} p_f(\theta_k) p_b(\theta_k) f_k(c_k, \theta_k)\, d\theta_k$

Tikhonov Algorithm (cont'd)

- If the messages $P_d(c_k)$ were the exact a posteriori probabilities of the code symbols, it would be $p_d(\theta_k) = p(r_k|\theta_k)$
- The pdf $p_d(\theta_k)$ is a linear combination of Gaussian pdf
- This pdf will be approximated with a single Gaussian pdf with the same mean and variance (approximation based on the first and second moment matching)
- From direct calculation

$$E\{r_k|\theta_k\} = \alpha_k e^{j\theta_k} \quad , \quad var\{r_k|\theta_k\} = 2\sigma^2 + \beta_k - |\alpha_k|^2$$

where

$$\alpha_k = \sum_{c\in\mathcal{C}} cP_d(c_k = c) \quad , \quad \beta_k = \sum_{c\in\mathcal{C}} |c|^2 P_d(c_k = c)$$

Hence

$$p_d(\theta_k) \simeq \exp\left\{-\frac{|r_k - \alpha_k e^{j\theta_k}|^2}{2\sigma^2 + \beta_k - |\alpha_k|^2}\right\} \propto \exp\left\{2\frac{\mathrm{Re}[r_k \alpha_k^* e^{-j\theta_k}]}{2\sigma^2 + \beta_k - |\alpha_k|^2}\right\}$$

Tikhonov Algorithm (cont'd)

- Substituting this approximation in the recursive integral equations for $p_f(\theta_k)$ and $p_b(\theta_k)$, we find that these pdfs may be expressed as (Tikhonov distribution)

$$p_f(\theta_k) \propto \exp\{\mathrm{Re}[a_{f,k} e^{-j\theta_k}]\} \quad , \quad p_b(\theta_k) \propto \exp\{\mathrm{Re}[a_{b,k} e^{-j\theta_k}]\}$$

and

$$a_{f,k} = \frac{a_{f,k-1} + 2\frac{r_{k-1}\alpha_{k-1}^*}{2\sigma^2+\beta_{k-1}-|\alpha_{k-1}|^2}}{1+\sigma_\Delta^2\left|a_{f,k-1} + 2\frac{r_{k-1}\alpha_{k-1}^*}{2\sigma^2+\beta_{k-1}-|\alpha_{k-1}|^2}\right|}$$

$$a_{b,k} = \frac{a_{b,k+1} + 2\frac{r_{k+1}\alpha_{k+1}^*}{2\sigma^2+\beta_{k+1}-|\alpha_{k+1}|^2}}{1+\sigma_\Delta^2\left|a_{b,k+1} + 2\frac{r_{k+1}\alpha_{k+1}^*}{2\sigma^2+\beta_{k+1}-|\alpha_{k+1}|^2}\right|}$$

- The solution of the integral equations is exact for $\sigma_\Delta = 0$ and involves a very good approximation in the case of a time-varying channel phase $\Rightarrow$ in practice, the only approximation is that on $p_d(\theta_k)$

Tikhonov Algorithm (cont'd)

- Finally,

$$P_u(c_k) \propto \exp\left\{-\frac{|c_k|^2}{2\sigma^2}\right\} \mathrm{I}_0\left(\left|a_{f,k} + a_{b,k} + \frac{r_k c_k^*}{\sigma^2}\right|\right)$$

Hence

$$P_d(c_k) \quad \Rightarrow \quad (\alpha_k, \beta_k) \quad \text{(mean and mean square value)}$$

$$(\alpha_k, \beta_k) \quad \Rightarrow \quad a_{f,k}, a_{b,k} \quad \text{(forw. and back. recursions)}$$

$$a_{f,k}, a_{b,k} \quad \Rightarrow \quad P_u(c_k) \quad \text{(probability update)}$$

System Model: Flat-Fading Channels

- Flat fading channel:

$$r_k = h_k c_k + n_k$$

where $\{h_k\}$ is a sequence of zero-mean complex Gaussian random variables with autocorrelation sequence $\mathrm{E}\{h_k h_{k-n}^*\} = J_0(2\pi f_D T n)$, where $f_D T$ is the normalized Doppler rate

- The fading process is approximated as AR(n):

$$h_{k+1} = \sum_{i=0}^{n-1} \rho_i h_{k-i} + \nu_k$$

where ν_k is a complex white Gaussian process and coefficients ρ_i can be computed from the fading autocorrelation sequence

- Example: AR(1)

$$h_k = \rho h_{k-1} + \nu_k$$

where $\rho = J_0(2\pi f_D T)$ and ν_k has a variance $1 - \rho^2$

Overall Factor Graph

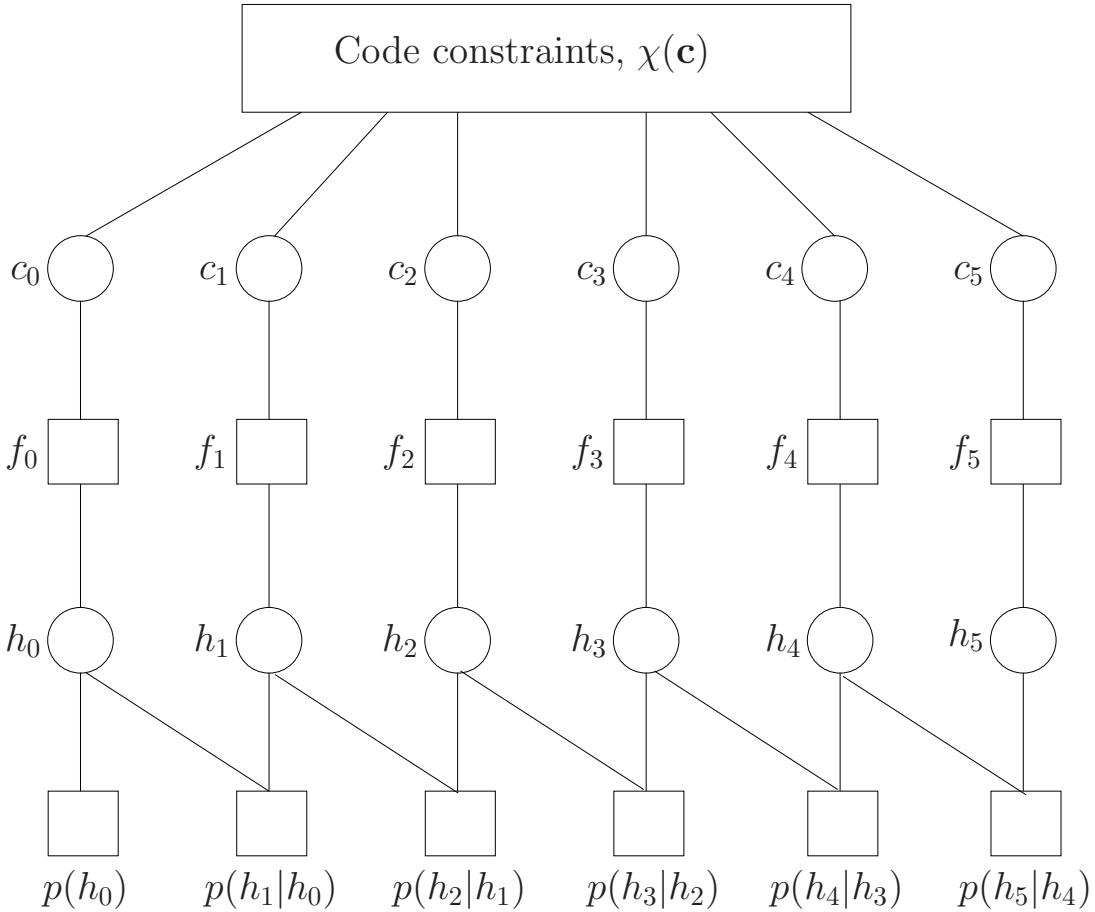
- The joint a posteriori distribution of symbols and unknown parameters may be expressed as

$$\begin{aligned} p(\mathbf{c},\mathbf{h}|\mathbf{r}) &\propto p(\mathbf{r}|\mathbf{c},\mathbf{h})\chi(\mathbf{c})p(\mathbf{h}) \\ &= \chi(\mathbf{c})p(h_0)\prod_{k=0}^{N-1} p(r_k|c_k,h_k)\prod_{k=1}^{N-1} p(h_k|h_{k-1}) \\ &\propto \chi(\mathbf{c})p(h_0)\prod_{k=0}^{N-1} f_k(c_k,h_k)\prod_{k=1}^{N-1} p(h_k|h_{k-1}) \end{aligned}$$

where

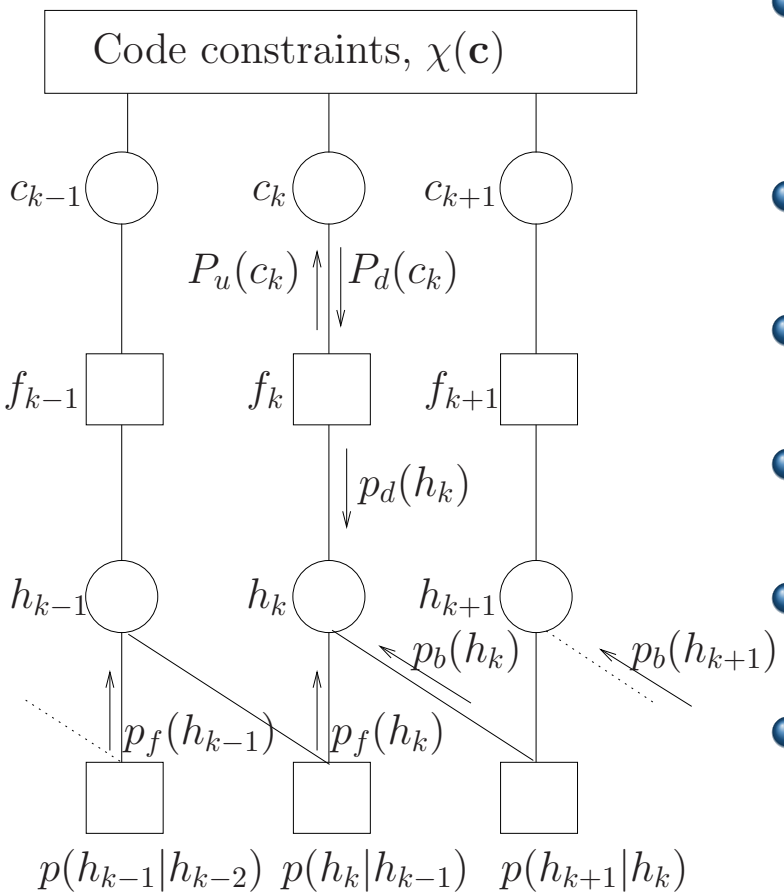
$$\begin{aligned} f_k(c_k,h_k) &= \exp\left\{-\frac{1}{2\sigma^2}|r_k - c_k h_k|^2\right\} \\ p(h_0) &= \frac{1}{\pi(1-\rho^2)} e^{-\frac{|h_0|^2}{1-\rho^2}} \\ p(h_k|h_{k-1}) &= \frac{1}{\pi(1-\rho^2)} e^{-\frac{|h_k - h_{k-1}|^2}{1-\rho^2}} \end{aligned}$$

Overall Factor Graph (cont'd)



- Corresponding FG

Overall Factor Graph (cont'd)



- Omitting the esplicit reference to the current iteration
- $p_d(h_k) = \sum_{c \in \mathcal{C}} P_d(c_k = c) f_k(c_k = c, h_k)$
- $p_f(h_k) = \int \int p_d(h_{k-1}) p_f(h_{k-1}) p(h_k|h_{k-1})\, dh_{k-1}$
- $p_b(h_k) = \int \int p_d(h_{k+1}) p_b(h_{k+1}) p(h_{k+1}|h_k)\, dh_{k+1}$
- $P_u(c_k) = \int \int p_f(h_k) p_b(h_k) f_k(c_k, h_k)\, dh_k$
- The optimal SP algorithm is unfeasible

Kalman Smoother

- The pdf $p_d(h_k)$ is a linear combination of Gaussian pdf
- This pdf will be approximated with a single Gaussian pdf with the same mean
- From direct calculation

$$E\{r_k|h_k\} = \alpha_k h_k$$

where

$$\alpha_k = \sum_{c \in \mathcal{C}} c P_d(c_k = c)$$

Hence

$$p_d(h_k) \simeq \exp\left\{-\frac{|r_k - \alpha_k h_k|^2}{2\sigma^2}\right\}$$

Kalman Smoother (cont'd)

- Substituting this approximation in the recursive integral equations for $p_f(h_k)$ and $p_b(h_k)$, we find that these pdfs are still Gaussian pdfs. Therefore, the problem reduces the propagation of their means and variances
- By using the following properties of Gaussian pdfs ($g_{\mathbb{C}}(A, \Sigma, x) = \frac{1}{\pi\Sigma} \exp\{-\frac{|x-A|^2}{\Sigma}\}$)

$$g_{\mathbb{C}}(A_1, \Sigma_1, x) g_{\mathbb{C}}(A_2, \Sigma_2, x) \propto g_{\mathbb{C}}\left(\frac{\Sigma_2}{\Sigma_1+\Sigma_2}A_1 + \frac{\Sigma_1}{\Sigma_1+\Sigma_2}A_2, \frac{\Sigma_1\Sigma_2}{\Sigma_1+\Sigma_2}, x\right)$$

$$\int g_{\mathbb{C}}(A_1, \Sigma_1, x) g_{\mathbb{C}}(A_2 x, \Sigma_2, y) dx \propto g_{\mathbb{C}}\left(A_1 A_2, \Sigma_2 + |A_2|^2 \Sigma_1, y\right)$$

we obtain the following forward and backward recursions

Kalman Smoother (cont'd)

- Forward recursion

$$
\begin{aligned}
m_{k|k} &= m_{k|k-1} + \frac{\Sigma_{k|k-1}\alpha_k^*}{|\alpha_k|^2\Sigma_{k|k-1} + 2\sigma^2}\left(r_k - \alpha_k m_{k|k-1}\right) \\
\Sigma_{k|k} &= \frac{2\sigma^2}{|\alpha_k|^2\Sigma_{k|k-1} + 2\sigma^2}\Sigma_{k|k-1} \\
m_{k+1|k} &= \rho m_{k|k} \qquad\qquad \Sigma_{k+1|k} = \rho^2\Sigma_{k|k} + 1 - \rho^2
\end{aligned}
$$

for $k = 0, \ldots, N-1$, with initial conditions $\Sigma_{0|-1} = 1$ and $m_{0|-1} = 0$

- Backward recursion

$$
\begin{aligned}
\mu_{k|k} &= \mu_{k|k+1} + \frac{\Xi_{k|k+1}\alpha_k^*}{|\alpha_k|^2\Xi_{k|k+1} + 2\sigma^2}\left(r_k - \alpha_k \mu_{k|k+1}\right) \\
\Xi_{k|k} &= \frac{2\sigma^2}{|\alpha_k|^2\Xi_{k|k+1} + 2\sigma^2}\Xi_{k|k+1} \\
\mu_{k-1|k} &= \rho \mu_{k|k} \qquad\qquad \Xi_{k-1|k} = \rho^2\Xi_{k|k} + 1 - \rho^2
\end{aligned}
$$

for $k = N-1, \ldots, 0$, with initial conditions $\Xi_{N-1|N} = 1$ and $\mu_{N-1|N} = 0$.

Kalman Smoother (cont'd)

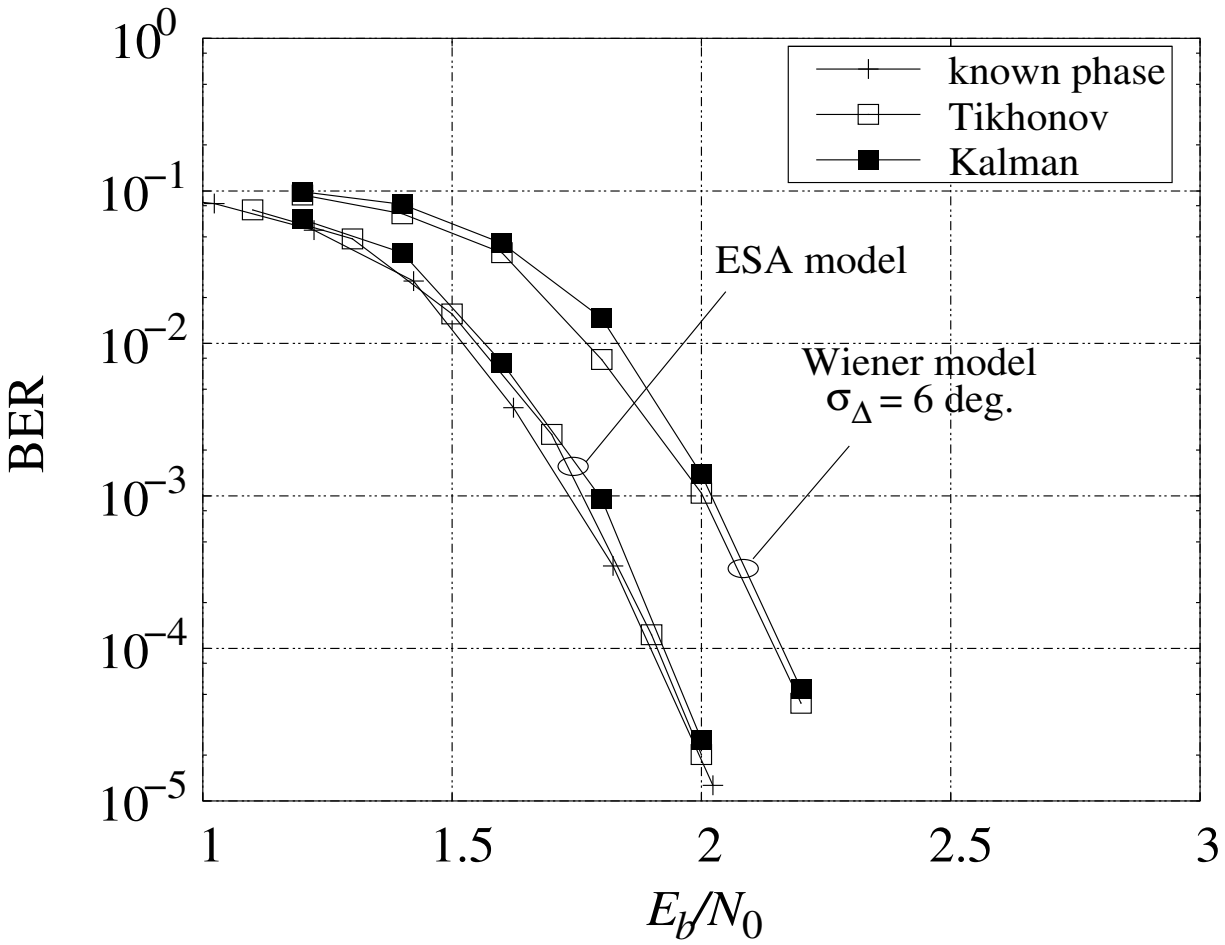
- Finally, for each k we obtain

$$
\begin{aligned}
m_k &= \frac{\Xi_{k|k+1}}{\Sigma_{k|k-1}+\Xi_{k|k+1}} m_{k|k-1} + \frac{\Sigma_{k|k-1}}{\Sigma_{k|k-1}+\Xi_{k|k+1}} \mu_{k|k+1} \\
\Sigma_k &= \frac{\Sigma_{k|k-1}\Xi_{k|k+1}}{\Sigma_{k|k-1}+\Xi_{k|k+1}}
\end{aligned}
$$

and

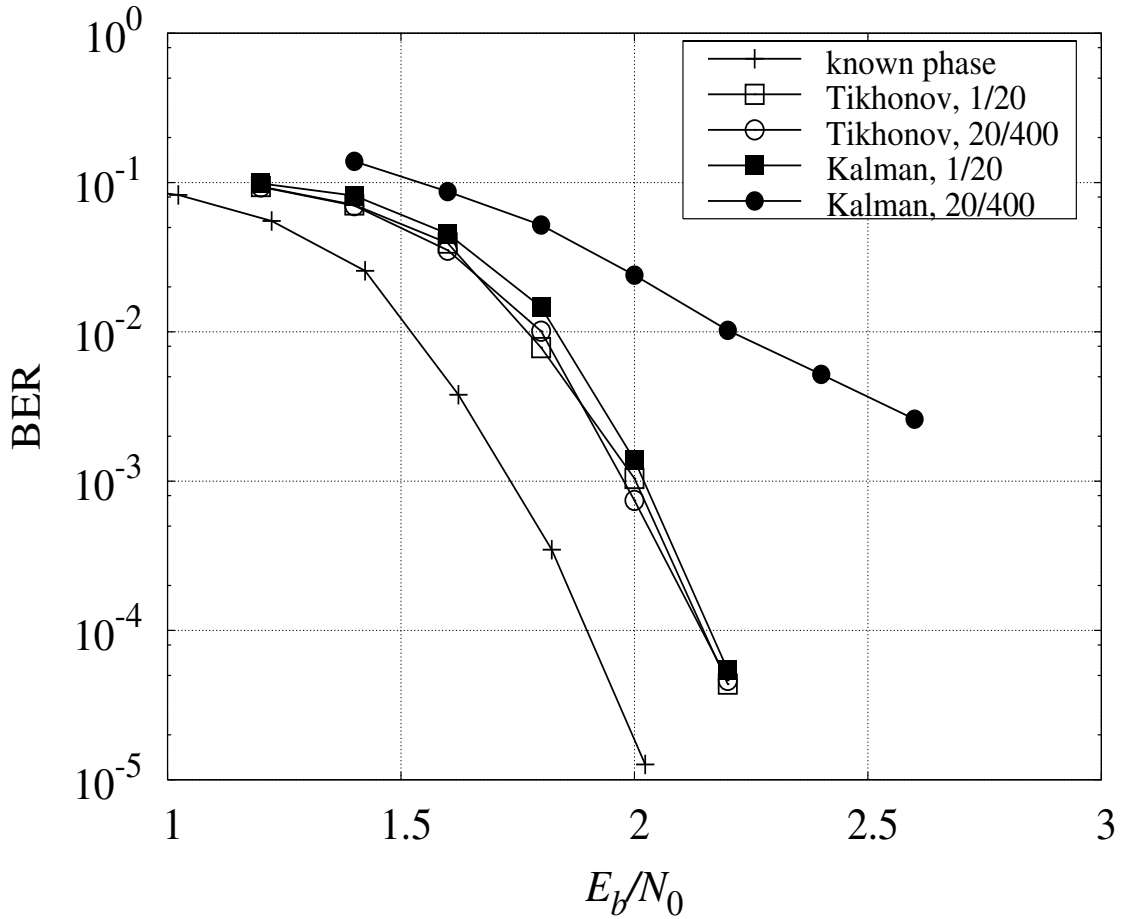
$$
P_u(c_k) \propto g_{\mathbb{C}}\left(m_k c_k, 2\sigma^2 + |c_k|^2 \Sigma_k, r_k\right)
$$

Numerical Results (1)



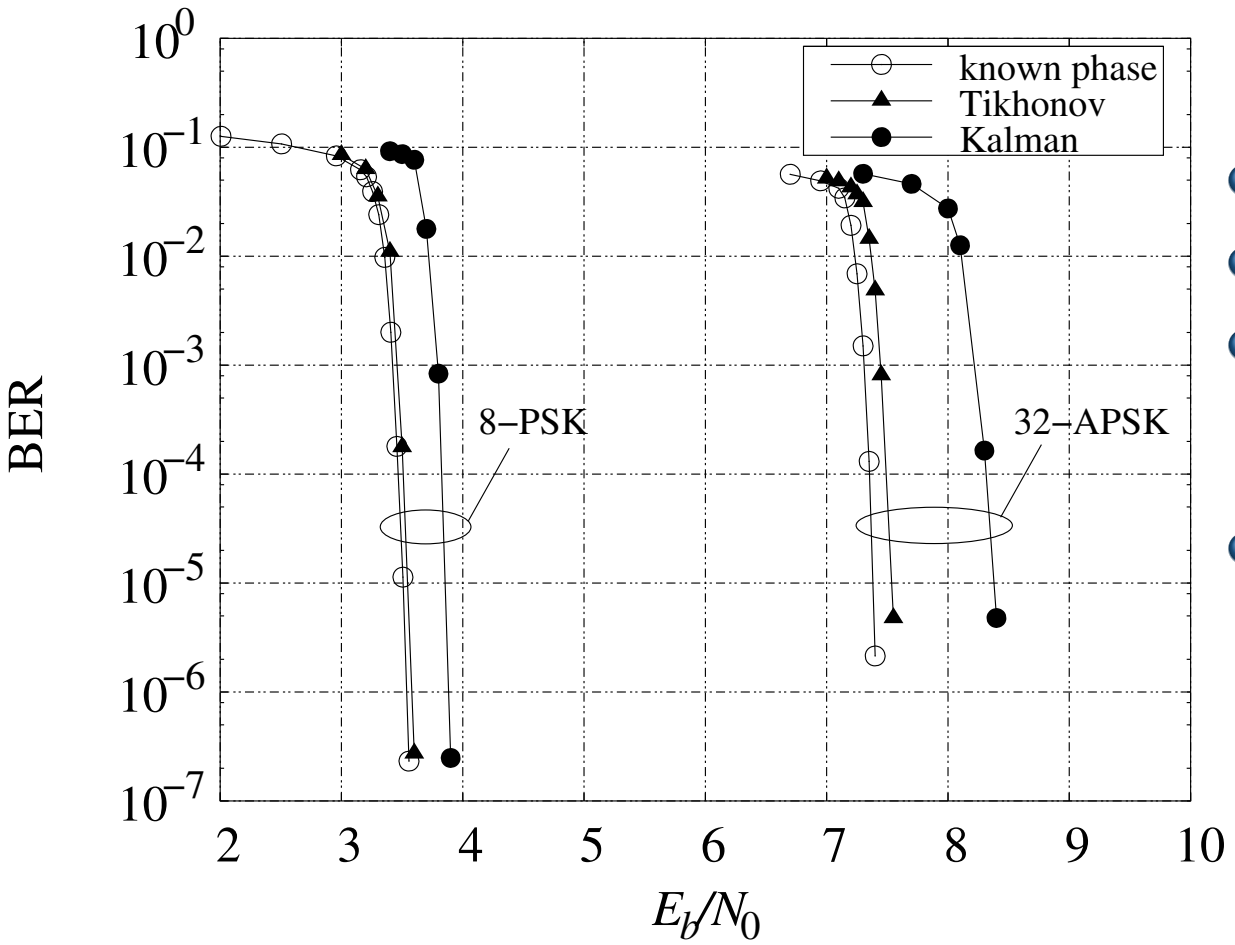
- (3,6)-regular LDPC code with codewords of length 4000
- BPSK modulation
- Maximum of 200 iterations of the SP algorithm on the overall graph
- A pilot symbol every 19 code bits to avoid ambiguity $\Rightarrow$ increase in the required SNR of 0.223 dB

Numerical Results (2)



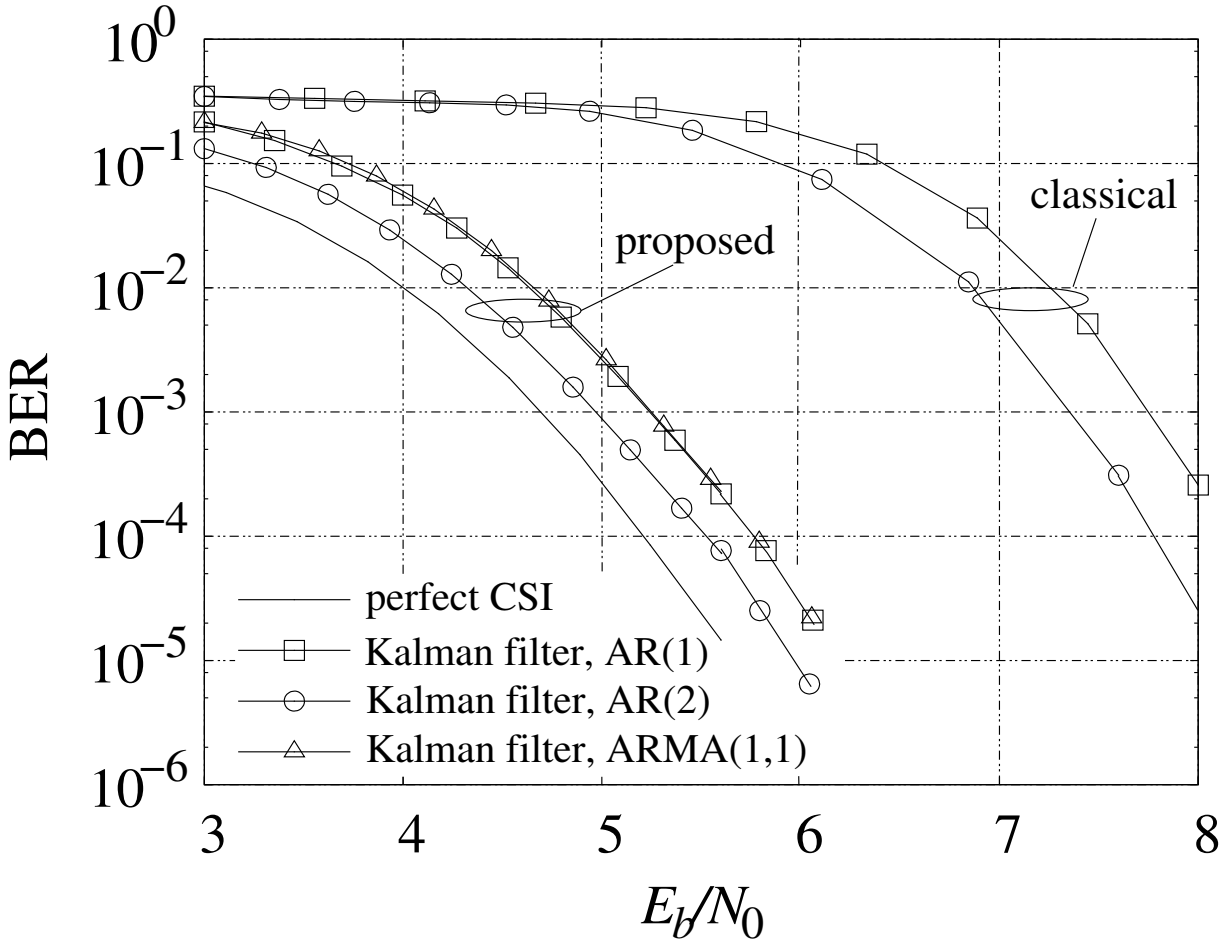
- Same code, modulation format, and number of pilots

Numerical Results (3)



- DVB-S2 phase model
- DVB-S2 LDPC codes
- Maximum of 50 iterations of the SP algorithm on the overall graph
- Pilot distribution of the DVB-S2 standard

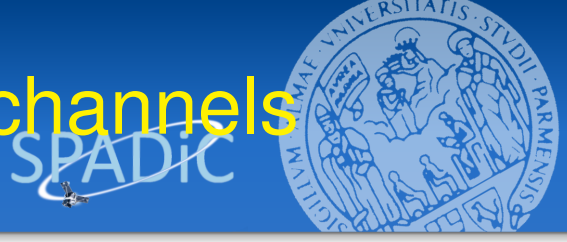
Numerical Results (4)



- Fading channel with $f_D T = 10^{-2}$
- (3,6)-regular LDPC code with codewords of length 4000
- BPSK modulation
- Maximum of 200 iterations of the SP algorithm on the overall graph
- A pilot symbol every 19 code bits to avoid ambiguity $\Rightarrow$ increase in the required SNR of 0.223 dB

Outline

Linear modulations over linear channels



- We consider linear modulations over linear channels impaired by AWGN
- The relationship between the transmitted sequence $\mathbf{a} = [c_1, c_2, \ldots, c_N]^T$ and the received sequence $\mathbf{r} = [r_1, r_2, \ldots, r_K]^T$ can be written as

$$\mathbf{r} = \mathbf{H}\mathbf{c} + \mathbf{w}$$

  where $\mathbf{w} = [w_1, w_2, \ldots, w_K]^T$ are i.i.d. zero-mean Gaussian random variables, while $\mathbf{H}$ is a matrix with K rows and N columns
- We consider MAP symbol detection of the symbols $\mathbf{c}$, that requires the evaluation of the APPs $P(c_n|\mathbf{r})$
- To be used in iterative detection/decoding schemes

Examples of Channels

- Channels with known ISI: the model holds with $N = K$ and

$$H_{m,n} = \begin{cases} f_{m-n} & \text{for } 0 \leq m-n \leq L \\ 0 & \text{otherwise} \end{cases}$$

  $\{f_\ell\}_{\ell=0}^{L}$ the discrete-time channel impulse response after WMF

- CDMA systems (only one symbol epoch): the matrix **H** includes the spreading sequence of the users and the related powers. N is the number of users and K is the length of the spreading sequences
- Multiple-antenna channels (only one symbol epoch): K and N represent the number of receive and transmit antennas, whereas the entries of matrix **H** are the channel coefficients.
- OFDM systems with ICI: the matrix **H** is a square matrix where the entry $H_{m,n}$ describes the ICI between the m-th and n-th subcarriers
- Spectrally-efficient FDM systems (systems with both ISI and ICI)
- Other multidimensional ISI channels (storage systems with bidimensional ISI)

Literature Review

- The literature addressing suboptimal soft-input soft-output (SISO) detection algorithms for applications that can be led to the mentioned system model is huge. Most papers address only one of these applications at a time
- Optimal MAP symbol detection can be performed with a complexity which increases exponentially with the number of interferers
- The most famous paper is that by Wang & Poor (Gaussian approximation of the interference) addressing multiuser detection for CDMA systems. It has then be extended to MIMO, ISI channels, FDM systems with ICI. The resulting algorithms have a complexity which increases quadratically with the number of interferers

New Algorithm for MAP Symbol Detection

The conditional pdf of the received sequence $\mathbf{y}$ given the modulation symbols $\mathbf{c}$ is

$$p(\mathbf{y}|\mathbf{c}) \propto \exp\left(-\frac{\|\mathbf{y}-\mathbf{Hc}\|^2}{2\sigma^2}\right)$$

If we define

$$\mathbf{x} = \mathbf{H}^H\mathbf{y}, \quad \mathbf{G} = \mathbf{H}^H\mathbf{H}$$

$$p(\mathbf{y}|\mathbf{c}) \propto \exp\left(\frac{2\mathrm{Re}\{\mathbf{c}^H\mathbf{x}\} - \mathbf{c}^H\mathbf{Gc}}{2\sigma^2}\right)$$

$\mathbf{x}$ is an equivalent sufficient statistic for MAP detection
We call $\mathbf{y}$ and $\mathbf{x}$ as Forney and Ungerboeck observation models.

$$\mathbf{x} = \mathbf{H}^H(\mathbf{Hc}+\mathbf{w}) = \mathbf{Gc}+\boldsymbol{\eta}$$

$$x_n = G_{n,n}c_n + \sum_{m\neq n} G_{n,m}c_m + \eta_n$$

When $\mathbf{G}$ is not diagonal, the computation of the target APPs has a complexity that grows exponentially with the number of interferers

Proposed Graph-Based Detection Algorithm

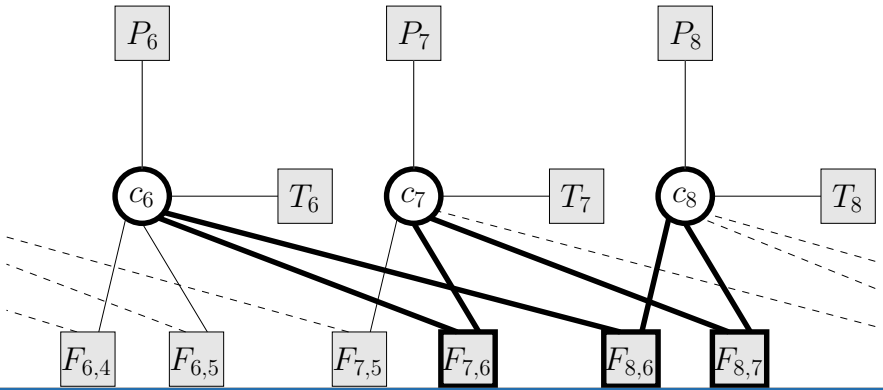
Novel framework for SISO detection algorithm, obtained by using the FG/SPA framework.

$$P(\mathbf{c}|\mathbf{y}) \propto P(\mathbf{c})\, p(\mathbf{y}|\mathbf{c}) \propto \prod_{n=1}^{N} \left[P_n(c_n) T_n(c_n) \prod_{m<n} F_{n,m}(c_n, c_m) \right]$$

$$T_n(c_n) = \exp\left[\frac{1}{\sigma^2} \mathrm{Re}\left\{ x_n c_n^* - \frac{G_{n,n}}{2} |c_n|^2 \right\} \right]$$

$$F_{n,m}(c_n, c_m) = \exp\left[-\frac{1}{\sigma^2} \mathrm{Re}\left\{ G_{n,m} c_m c_n^* \right\} \right]$$

Three sections of the FG, for the case when interference between c_n and c_m arises only if $|m-n| \in \{1,2\}$

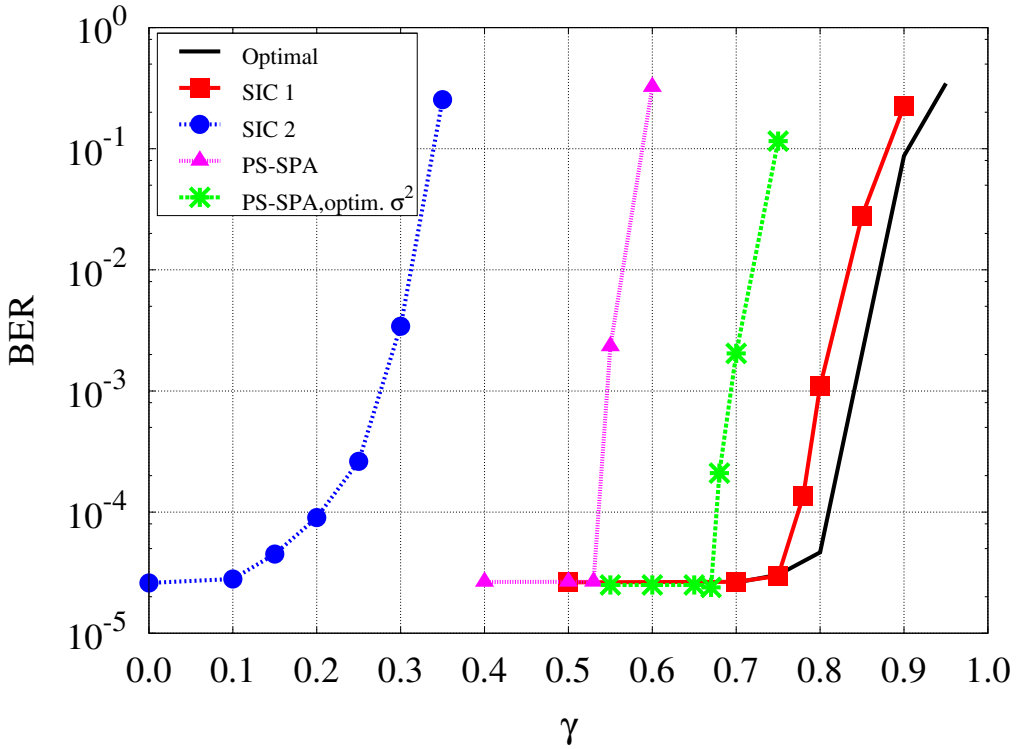


Numerical Results: CDMA Systems

CDMA system with $N = 4$ synchronous users and $\mathrm{SNR} = 1.35$ dB

Constant cross-correlation of the spreading sequences,

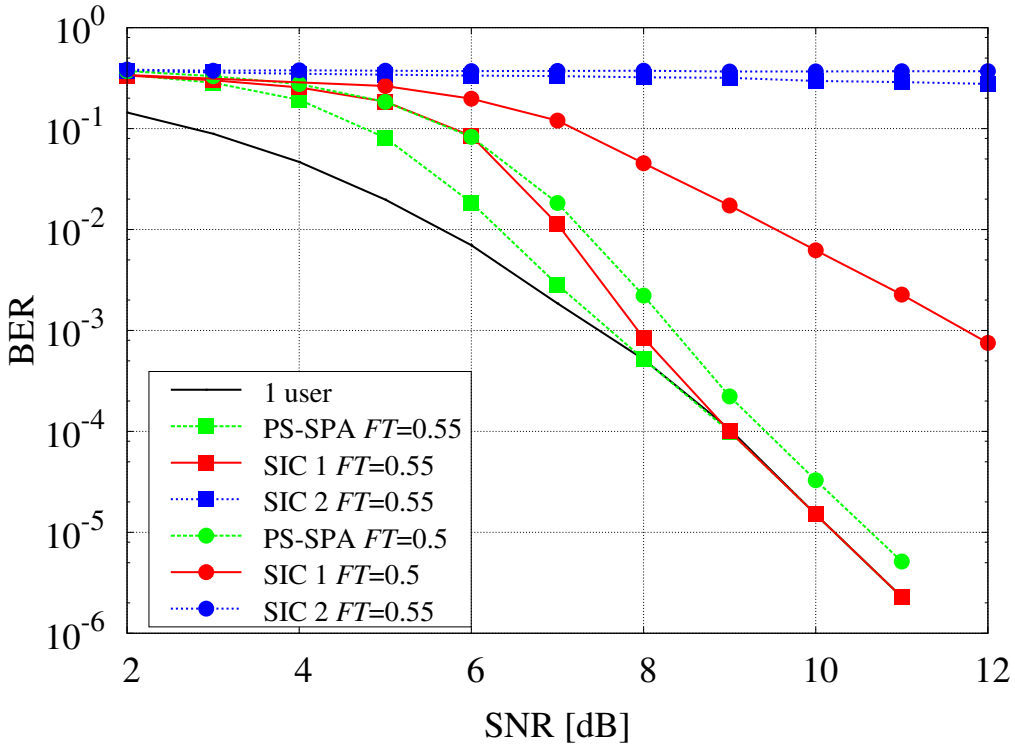
$$G_{n,m}/G_{n,n} = \gamma \in (0, 1)$$



- rate-$1/2$ CC with generators $(23, 35)_8$, a BPSK modulation
- randomly-generated bit interleaver of length 256 bits
- 15 iterations between detector and decoder

Numerical Results: FDM Systems

FDM systems with $U = 3$ synchronous users for different normalized channel spacings $FT = |f^{(i)} - f^{(i-1)}|T$

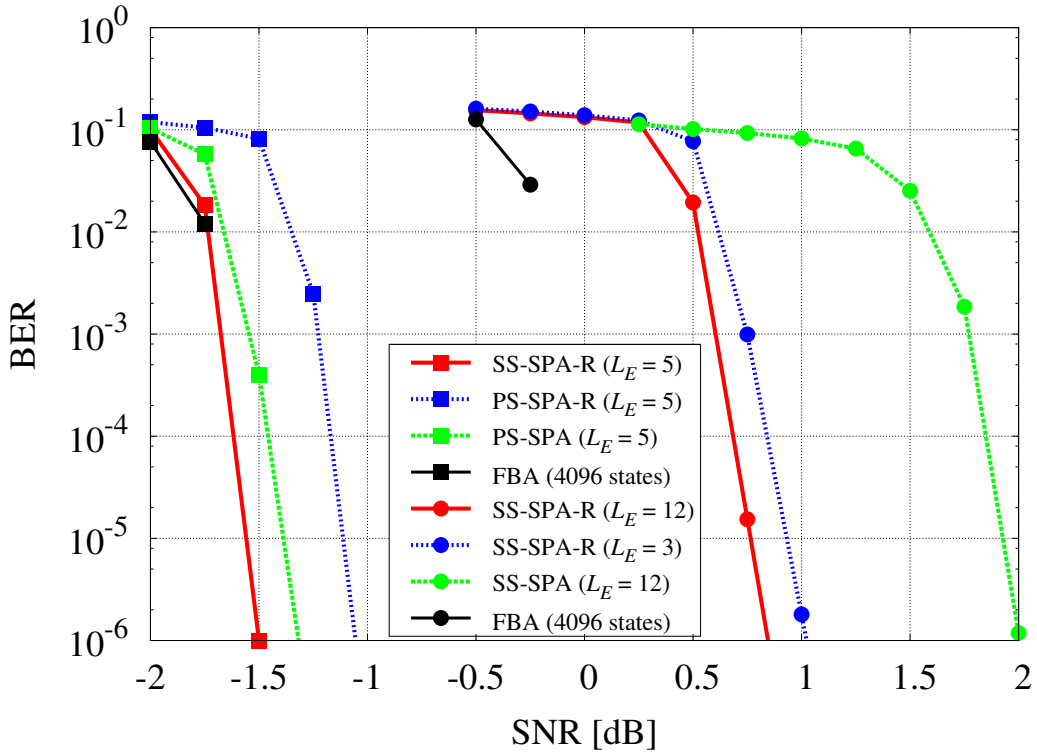


- rate-1/2 CC with generators $(23, 35)_8$, a 8-PSK modulation
- randomly-generated bit interleaver of lenght 512
- 10 iterations between detector and decoders

Numerical Results: ISI Channels

Two channels with ($L = 12$)

The optimal BCJR algorithm works on a trellis with 4096 states when a binary modulation is considered



- Outer rate-1/2 LDPC decoder
- FTN (RRC with α=0.1 and $\tau = 0.78$)
- Magnetic channel with $D = 3$

Outline

Conclusions

- Factor graphs provide a natural graphical description of the factorization of a global function into a product of local functions
- Factor graphs can be applied in a wide range of application areas (from communications and signal processing to neural networks)
- The sum-product algorithm represents a general method to marginalize the global function
- Some applications to communications have been shown
- This framework can be successful applied to other communication problems: the work has just started!

References

- S. M. Aji and R. J. McEliece, “The generalized distributive law,” IEEE Transactions on Information Theory, vol. 46, pp. 325-343, Feb. 2000
- F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” IEEE Trans. Inform. Theory, vol. 47, pp. 498-519, Feb. 2001
- A. P. Worthen and W. E. Stark, “Unified design of iterative receivers using factor graphs,” IEEE Trans. Inf. Theory, vol. 47, pp. 843-849, Feb. 2001
- J. Boutros and G. Caire, “Iterative multiuser joint decoding: unified framework and asymptotic analysis,” IEEE Trans. Inform. Theory, vol. 48, pp. 1772-1793, July 2002
- G. Colavolpe and G. Germi, “On the application of factor graphs and the sum-product algorithm to ISI channels,” IEEE Trans. Commun., vol. 53, pp. 818-825, May 2005
- H. Niu, M. Shen, J. A. Ritcey, and H. Liu, “A factor graph approach to iterative channel estimation and LDPC decoding over fading channels,” IEEE Trans. Wireless Commun., vol. 4, no. 4, pp. 1345-1350, July 2005
- G. Colavolpe, A. Barbieri, and G. Caire, “Algorithms for iterative decoding in the presence of strong phase noise,” IEEE J. on Selected Areas Commun., vol. 23, pp. 1748-1757, September 2005
- G. Colavolpe, “On LDPC codes over channels with memory,” IEEE Trans. Wireless Commun., vol. 5, pp. 1757-1766, July 2006