ELG5131 Project 1

Name: Siqi Ma

Student number: 7070144

1. Brief introduction of the project.

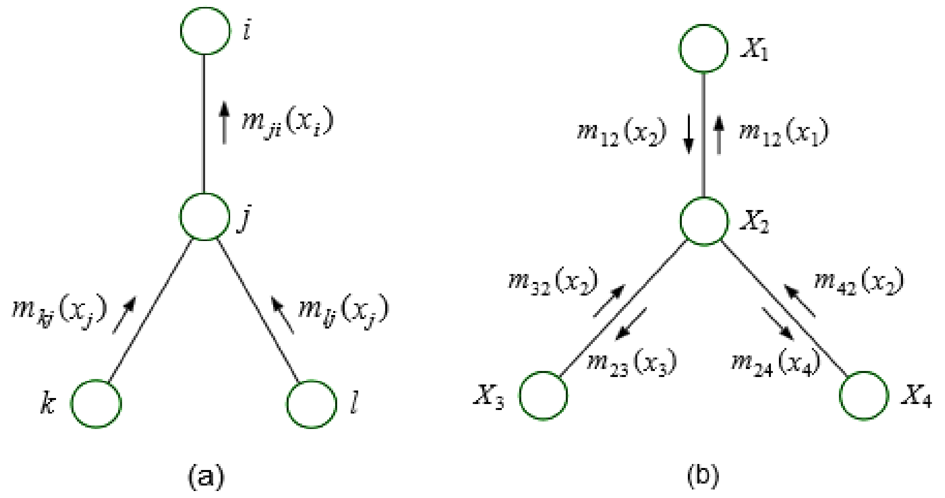
In this project, we should build a communication system which uses two kinds of decoding algorithms. The algorithms are Sum-Product and Max-Product Decoding. Given fixed σ^2, compare the two ways in the aspect of bit error probability. Then make a conclusion of the two algorithms.

2. The algorithms used in the project

2.1 Sum-Product decoding

Sum-product algorithm operates in a factor graph and attempts to compute various marginal functions associated with the global function.

Basic idea:



The desired marginal is obtained as

$$m_{ji}(x_i) = \sum_{x_j} \varphi(x_j)\varphi(x_i, x_j) \prod_{x_k \in N(j)\backslash i} m_{kj}(x_j)$$

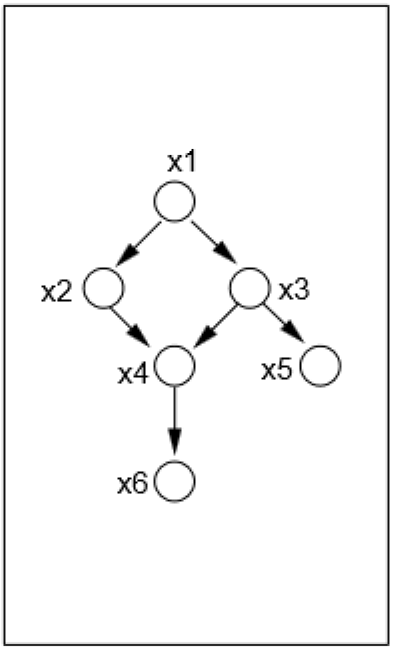
Where N(j) is the set of neighbors of node j, and where $m_{ji}(x_i)$is the intermediate term that is created when node j is eliminated

The desired marginal is obtained as

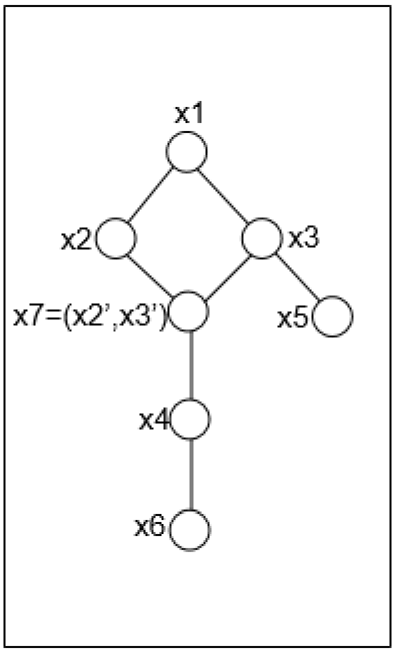
$$P(x_f) \propto \varphi(x_f) \prod_{e \in N(f)} m_{ef}(x_f)$$

2.2 Max-Product Decoding

The max-product "belief propagation" algorithm is a local-message passing algorithm on this graph that is known to converge to a unique fixed point when the graph is a tree.



a



b

$$\mathrm{m}_{\mathrm{i,j}}(\mathrm{x}_\mathrm{j}) \leftarrow \mathrm{a}\max_{\mathrm{x}_\mathrm{i}} \varphi_{\mathrm{ij}}(x_i, x_j) m_{ii}(x_i) \prod_{\mathrm{x}_\mathrm{k} \in \mathrm{N}(\mathrm{x}_\mathrm{i}) \backslash \mathrm{x}_\mathrm{j}} \mathrm{m}_{\mathrm{ki}}(\mathrm{x}_\mathrm{i})$$

$$\mathrm{b}_\mathrm{i}(\mathrm{x}_\mathrm{i}) \leftarrow \mathrm{a} m_{ii}(x_i) \prod_{\mathrm{x}_\mathrm{k} \in \mathrm{N}(\mathrm{x}_\mathrm{i})} \mathrm{m}_{\mathrm{ki}}(\mathrm{x}_\mathrm{i})$$

In which x_i, x_j are two neighboring nodes. $m_{ii}(x_i)$ is the message y_i send to x_i, and by $\mathrm{b}_\mathrm{i}(\mathrm{x}_\mathrm{i})$ the belief at node x_i

2.3 (7, 4) Hamming code:

From the equations:

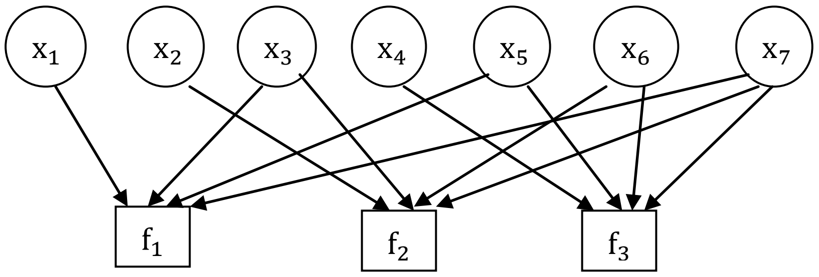
$$\mathrm{x}_4 + \mathrm{x}_5 + \mathrm{x}_6 + \mathrm{x}_7 = 0 \bmod 2$$

$$\mathrm{x}_2 + \mathrm{x}_3 + \mathrm{x}_6 + \mathrm{x}_7 = 0 \bmod 2$$

$$\mathrm{x}_1 + \mathrm{x}_3 + \mathrm{x}_5 + \mathrm{x}_7 = 0 \bmod 2$$

If we assume the function nodes are f_i , i=1, 2, 3

The following graph can be plot:



$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \text{ and } \mathrm{XH}^\mathrm{T} = 0$$

In which we use f_1, f_2, f_3 as exclusive or function.

3. The result of the project

3.1. About the programming

To calculate the bit error rate:

$$\text{BER}=\frac{the\ number\ of\ error\ bits}{the\ number\ of\ whole\ bits}$$

In case of getting the BER，we should get the number of the bits which are not successfully transmitted.

If the transmitted number is larger than 0, we decided it as 1 , which means that the bit of the send message is 0. Otherwise, we considered the bit of the send message as 1. So when we send an 1, the transmitted message is -1; if the Gaussian noise is larger than +1, an error will occur. If the bit of the send message is 0, if the noise is less than -1, an error will occur.

So the bit error rate can be calculated as follows

1) If the codeword is larger than 0:

$$P=\frac{1}{1+e^{-2\times x\div\sigma^2}}$$

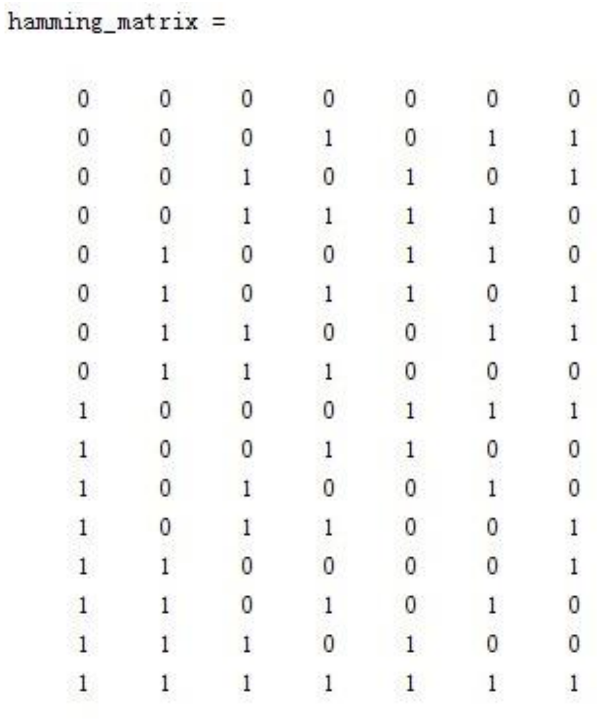
2) If the codeword is less than 0, then

$$P=\frac{1}{1+e^{2\times x\div\sigma^2}}$$

In which x is the element in the transmit signal.

3.2. Result and conclusion

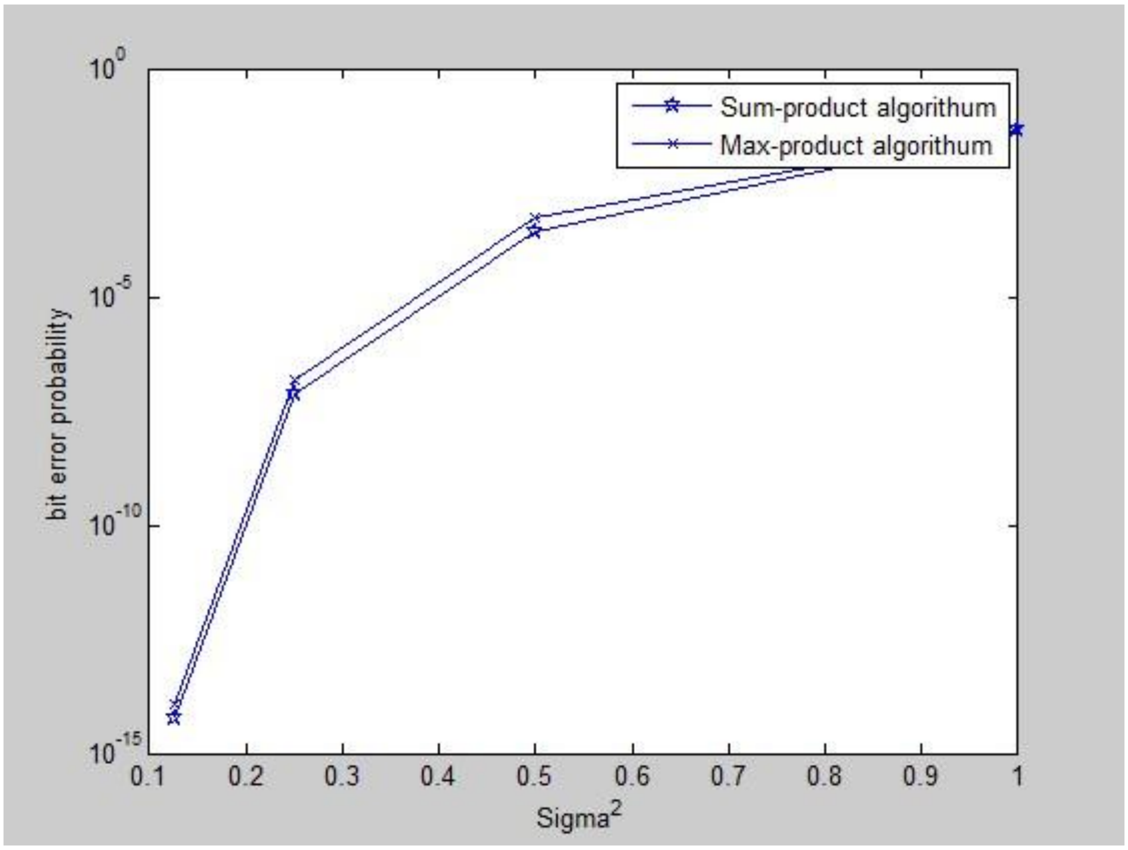
3.2.1 hamming 7,4 code



```
hamming_matrix =

     0     0     0     0     0     0     0
     0     0     0     1     0     1     1
     0     0     1     0     1     0     1
     0     0     1     1     1     1     0
     0     1     0     0     1     1     0
     0     1     0     1     1     0     1
     0     1     1     0     0     1     1
     0     1     1     1     0     0     0
     1     0     0     0     1     1     1
     1     0     0     1     1     0     0
     1     0     1     0     0     1     0
     1     0     1     1     0     0     1
     1     1     0     0     0     0     1
     1     1     0     1     0     1     0
     1     1     1     0     1     0     0
     1     1     1     1     1     1     1
```

The result of the program



For Gaussians, max-product and sum-product are identical. It is obvious the max-product algorithm is more efficient than the sum-product algorithm.

It is easy to show that sum-product belief propagation gives the turbo decoding algorithm and max-product belief propagation gives the modified turbo decoding algorithm

Appendix

Reference:

1. Factor Graphs and the Sum-Product Algorithm, Frank R. Kschischang, Senior Member, IEEE,February,2001

2. On the fixed points of the max-product algorithm, William T. Freeman, Yair Weissy, January 2000

3. Design of Hamming code based on MATLAB, Hao Xiao , Wang Xiaofeng

http://wenku.baidu.com/view/2745277701f69e314332944c.html

Code:

proj1.m

```
transmit_signal = zeros(16,7);
noise = zeros(16,7);
ts = zeros(16,7);
H = [1 1 1 0 1 0 0; 1 1 0 1 0 1 0; 1 0 1 1 0 0 1] %define parity-check
matrix H
msg = [0 0 0 0; 0 0 0 1; 0 0 1 0; 0 0 1 1; 0 1 0 0; 0 1 0 1;...
    0 1 1 0; 0 1 1 1; 1 0 0 0; 1 0 0 1; 1 0 1 0;1 0 1 1;...
    1 1 0 0; 1 1 0 1; 1 1 1 0; 1 1 1 1];      %define message signal
G = gen2par(H);        %generate the generator matrix of x
hamming_matrix = rem(msg* G,2)       %hamming 7,4 of the matrix
transmit_signal = 1 - 2*hamming_matrix;
n = randn(16,7);    %produce gaussian noise
ts=n+transmit_signal;            %add noise to signal
bit=1;
[a,b]=size(ts)
sigma2=[1 0.5 0.25 0.125];
column=1;
k=7;
bitnum=1;
%calculate the bit error rate of the two algorithms.
for i=1:1:4
    for j=1:1:k
       if transmit_signal(column,j)<0     %the BER when the received
signal is less than 0
          p(1,j)=1/(1+exp(2*ts(1,j)/sigma2(1,i)));
          if bit==1
             bitnum=j;
          end
       end
       if transmit_signal(column,j)>0      %the BER when the received
signal is greater than 0
          p(1,j)=1/(1+exp(-2*ts(1,j)/sigma2(1,i)));
          if bit==0
             bitnum=j;
          end
       end

%Sum-product algorithum
P(i,1)=p(1,bitnum);
P(i,1)=1-P(i,1);
```

```
%Max-product algorithum
P2(i,1)=p(1,1);
for k=1:1:n
    P2(i,1)=P2(i,1)*p(1,k);
end
P2(i,1)=1-P2(i,1);
    end
end
%plot the picture of the BER.
semilogy(sigma2,P,'bp-');
hold on
semilogy(sigma2,P2,'bx-');
legend('Sum-product algorithum','Max-product algorithum')
xlabel('Sigma^2');ylabel('bit error probability');
```