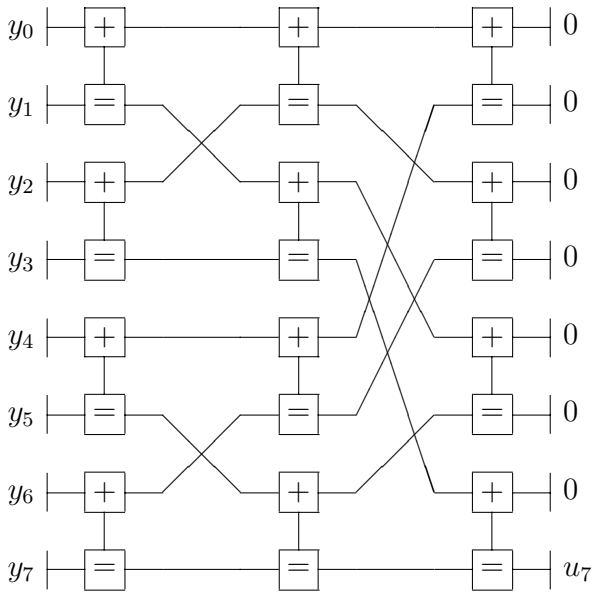
Problem Set 8 Solutions (revised)

Problem 8.1 (Realizations of repetition and SPC codes)

Show that a reduced Hadamard transform realization of a repetition code $\mathrm{RM}(0, m)$ *or a single-parity-check code* $\mathrm{RM}(m-1, m)$ *is a cycle-free tree-structured realization with a minimum number of* $(3, 1, 3)$ *repetition constraints or* $(3, 2, 2)$ *parity-check constraints, respectively, and furthermore with minimum diameter (distance between any two code symbols in the tree).*

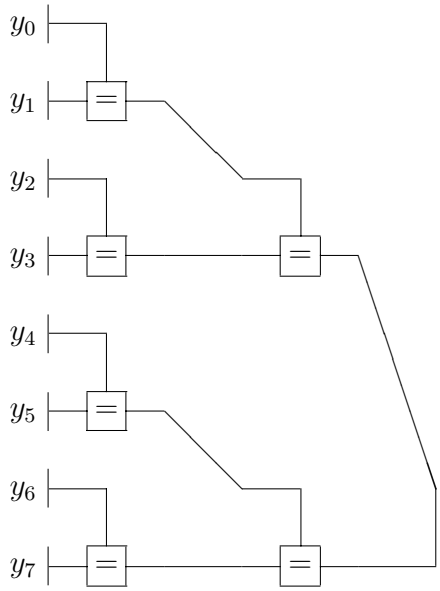
A Hadamard transform (HT) realization of a repetition code $\mathrm{RM}(0, m)$ of length 2^m is $\mathbf{y} = \mathbf{u}U_m$, where all components of $\mathbf{u}$ are zero except the last one, u_{2^m-1}, which multiplies the all-one (last) row of the universal generator matrix U_m. For example, the HT realization of the $(8, 1, 8)$ repetition code is shown below.



HT realization of $(8, 1, 8)$ RM code.

This realization may be simplified by the following two types of reductions:

This yields the reduced realization shown below.



Reduced HT realization of $(8, 1, 8)$ repetition code.

The reduced realization in this case is a binary tree-structured realization with 6 $(3, 1, 3)$ repetition constraints. In general, it is a binary tree-structured realization with $2^m - 2$ $(3, 1, 3)$ repetition constraints.

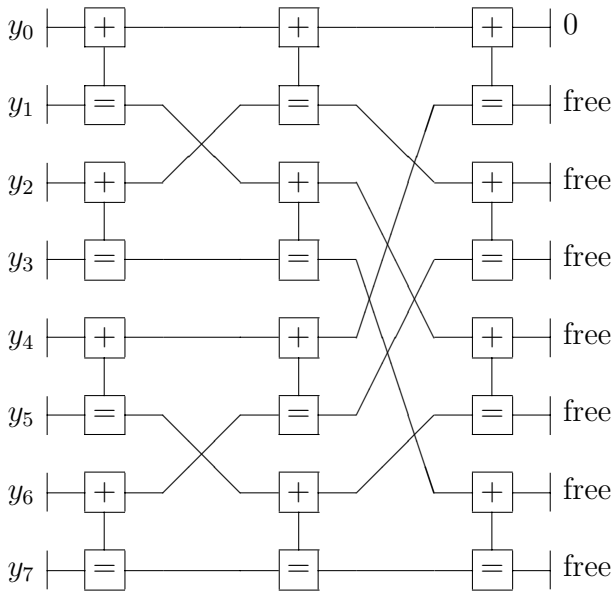
It is not hard to see that in general, an $(n, 1, n)$ repetition code may be realized by connecting $n - 2$ $(3, 1, 3)$ repetition constraints together in any cycle-free manner. There will then be n external variables of degree 1 and $n - 3$ internal variables of degree 2, with total degree equalling the total degree $3(n - 2)$ of the repetition constraints.

Since $n - \delta$ nodes require at least $n - \delta - 1$ edges to form a connected graph, and $3(n - \delta) - n < 2(n - \delta - 1)$ if $\delta > 2$, the minimum possible number of $(3, 1, 3)$ repetition constraints is $n - 2$.

The diameter of this graph is 3 (not counting half-edges). More generally, the diameter of a binary tree-structured graph like this with $2^m - 2$ nodes is $2m - 3$. It is not hard to see that any different method of interconnecting $2^m - 2$ nodes will increase the diameter.

*****

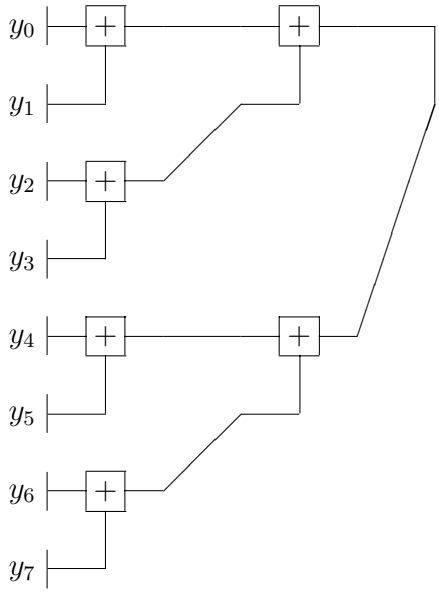
Similarly, for a HT realization of a single-parity-check code $\mathrm{RM}(m-1, m)$ of length 2^m, all components of $\mathbf{u}$ are free except the first one, u_0, which multiplies the weight-one (first) row of the universal generator matrix U_m. For example, the HT realization of the $(8, 1, 8)$ repetition code is shown below.



HT realization of $(8, 7, 2)$ RM code.

This realization may be simplified by the following two types of reductions:

For example, a reduced HT realization of the $(8, 7, 2)$ single-parity-check code is shown below.



Reduced HT realization of $(8, 7, 2)$ single-parity-check code.

This realization evidently has the same number of constraints and the same diameter as the reduced realization of the $(8, 1, 8)$ code above, and minimality is shown by the same arguments.

Show that these two realizations are duals; i.e., one is obtained from the other via interchange of $(3, 2, 2)$ *constraints and* $(3, 1, 3)$ *constraints.*

Obvious.

Problem 8.2 (Dual realizations of RM codes)

Show that in general a Hadamard transform (HT) realization of any Reed-Muller code $\mathrm{RM}(r, m)$ *is the dual of the HT realization of the dual code* $\mathrm{RM}(m - r - 1, m)$*; i.e., one is obtained from the other via interchange of* $(3, 2, 2)$ *constraints and* $(3, 1, 3)$ *constraints.*

As announced in class, you do not have to do this problem because the problem statement is incomplete, and you do not have all the background that you need to complete the proof. Nonetheless, let us sketch the proof.

The problem statement is incomplete in that in the dual realization, we must also exchange zero variables and free variables (which may be considered to be dual $(1, 0, \infty)$ and $(1, 1, 1)$ constraints, respectively). Also, we must invert the graph (take the top to bottom mirror image). Note that inversion of the graph replaces all constraints by their dual constraints, and complements the binary expansion of the indices of the components of $\mathbf{y}$ and $\mathbf{u}$.

The code $\mathrm{RM}(r, m)$ has the HT realization $\mathbf{y} = \mathbf{u}U_m$, where the components of $\mathbf{u}$ are free in the $k(r, m)$ positions corresponding to rows of U_m of weight 2^{m-r} or greater, and zero in the remaining positions. It is easy to see that these are the positions whose indices have m-bit binary expansions of weight $m - r$ or greater. For example, for the $(8, 1, 8)$ code $\mathrm{RM}(0, 3)$, only u_7 is free; for the $(8, 4, 4)$ code $\mathrm{RM}(1, 3)$, u_3, u_5, u_6 and u_7 are free; and for the $(8, 7, 2)$ code $\mathrm{RM}(2, 3)$, all but u_0 are free.

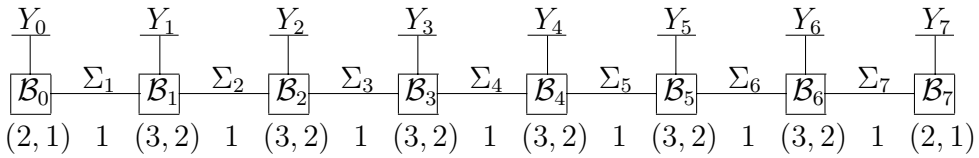
The indices of the rows of U_m of weight less than 2^{m-r} are thus the $2^m - k(r, m)$ indices whose m-bit binary expansions have weight less than $m - r$. The complements of these m-bit indices thus are the indices whose m-bit expansions have weight greater than r, or equivalently weight at least $r + 1$. These are precisely the indices of the free components of $\mathbf{u}$ for the code $\mathrm{RM}(m-r-1, m)$. Thus the zero components of $\mathbf{u}$ for the HT realization of $\mathrm{RM}(r, m)$ transform under the graph inversion into the free components of $\mathbf{u}$ for the HT realization of $\mathrm{RM}(m - r - 1, m)$, and *vice versa*.

Since dual normal graphs realize dual codes, this argument proves that $\mathrm{RM}(r, m)$ and $\mathrm{RM}(m - r - 1, m)$ are dual codes, with $k(m - r - 1, m) = 2^m - k(r, m)$.

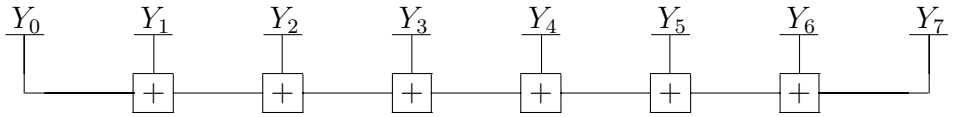
Problem 8.3 (revised) (BCJR (sum-product) decoding of SPC codes)

As shown in Problem 6.4 or Figure 1 of Chapter 10, any $(n, n-1, 2)$ *single-parity-check code has a two-state trellis diagram. Consider the* $(8, 7, 2)$ *code, and let the received sequence from a discrete-time AWGN channel with input alphabet* $\{\pm 1\}$ *and noise variance* $\sigma^2 = 1$ *be given by* $\mathbf{r} = (0.1, -1.0, -0.7, 0.8, 1.1, 0.3, -0.9, 0.5)$. *Perform BCJR (sum-product) decoding of this sequence to determine the APP vector for each symbol* Y_k. *[Hint: the special algorithm given in Section 13.5.2 (see Problem 9.5) to implement the sum-product update rule for zero-sum nodes may be helpful here.]*

First, let us draw the normal graph of a minimal trellis realization of the $(8, 7, 2)$ code. This graph is shown abstractly below:



Moreover, it is easy to see that the $(3, 2)$ branch constraint codes are all $(3, 2, 2)$ zero-sum codes, and the $(2, 1)$ codes are simple repetition codes that need not actually be shown. Therefore the trellis realization may be drawn simply as follows:



Note that this trellis realization of the $(8, 7, 2)$ code is another cycle-free realization that uses 6 $(3, 2, 2)$ zero-sum constraints, as in the reduced HT realization of Problem 8.1; however, the diameter of this realization is 5 (which is as large as it could possibly be).

For a binary-input Gaussian-noise channel with inputs $\{\pm 1\}$ and Gaussian conditional probability density $p(r \mid y) = (2\pi\sigma)^{-1/2} \exp -(r-y)^2/2\sigma^2$, the *a posteriori* probability (APP) of $y \in \{\pm 1\}$ given a received value r is, by Bayes' rule,

$$p(y \mid r) = \frac{p(r \mid y)}{p(r \mid 1) + p(r \mid -1)} = \frac{e^{ry/\sigma^2}}{e^{r/\sigma^2} + e^{-r/\sigma^2}}.$$

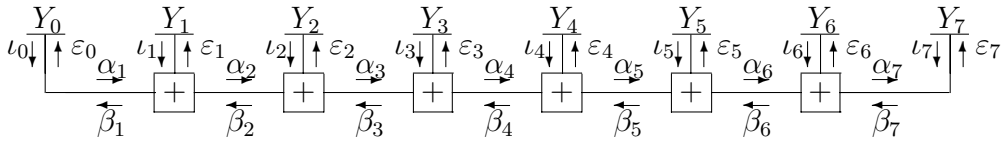
Here $\sigma^2 = 1$ (so SNR = 1 (0 dB); *i.e.,* the channel is very noisy).

The two values $p(\pm 1 \mid r_k)$ form the "intrinsic information" message $\iota_k = \{p_{0k}, p_{1k}\}$ derived from each received symbol $r_k, 0 \le k \le 7$. These values are computed from the received vector $\mathbf{r} = (0.1, -1.0, -0.7, 0.8, 1.1, 0.3, -0.9, 0.5)$ in the following table. (Note that it would have sufficed to use the unnormalized pair $\{e^{r_k/\sigma^2}, e^{-r_k/\sigma^2}\}$.) We also compute the Hadamard transform $\{p_{0k} + p_{1k} = 1, p_{0k} - p_{1k} = I_k\}$ of each pair of values for later use.

r_k	p_{0k}	p_{1k}	I_k	
$r_0 = +0.1$	0.55	0.45	1	$+0.10$
$r_1 = -1.0$	0.12	0.88	1	-0.76
$r_2 = -0.7$	0.20	0.80	1	-0.60
$r_3 = +0.8$	0.83	0.17	1	$+0.66$
$r_4 = +1.1$	0.90	0.10	1	$+0.80$
$r_5 = +0.3$	0.65	0.35	1	$+0.30$
$r_6 = -0.9$	0.14	0.86	1	-0.72
$r_7 = +0.5$	0.73	0.37	1	$+0.46$

Note that $I_k = \tanh r_k/\sigma^2$, so $I_k \approx r_k/\sigma^2$ for small r_k. The sign of I_k represents a "hard decision," whereas the magnitude $0 \le |I_k| \le 1$ represents the reliability of that decision.

We then propagate the APP messages through the graph below, using the BCJR (sum-product) algorithm. Note that in the forward direction $\alpha_1 = \iota_0$, α_k is the sum-product update of α_{k-1} and ι_{k-1} for $2 \le k \le 7$, and finally $\varepsilon_7 = \alpha_7$. Similarly, in the backward direction, $\beta_7 = \iota_7$, β_k is the sum-product update of β_{k+1} and ι_k for $6 \ge k \ge 1$, and finally $\varepsilon_0 = \beta_1$. Then ε_k is the sum-product update of α_k and β_{k+1} for $1 \le k \le 6$.



For a $(3, 2, 2)$ constraint code $\mathcal{C}_k$, the set of past 2-tuples consistent with a 0 value for any incident variable is $\mathcal{C}_k(0) = \{00, 11\}$, and similarly $\mathcal{C}_k(1) = \{01, 10\}$. Therefore the sum-product update rule is

$$\begin{aligned} p(0 \mid \mathbf{r}_{|\mathcal{P}}) &= p(0 \mid \mathbf{r}_{|\mathcal{P}_{j'}})p(0 \mid \mathbf{r}_{|\mathcal{P}_{j''}}) + p(1 \mid \mathbf{r}_{|\mathcal{P}_{j'}})p(1 \mid \mathbf{r}_{|\mathcal{P}_{j''}}); \\ p(1 \mid \mathbf{r}_{|\mathcal{P}}) &= p(0 \mid \mathbf{r}_{|\mathcal{P}_{j'}})p(1 \mid \mathbf{r}_{|\mathcal{P}_{j''}}) + p(0 \mid \mathbf{r}_{|\mathcal{P}_{j'}})p(1 \mid \mathbf{r}_{|\mathcal{P}_{j''}}), \end{aligned}$$

where $\mathcal{P}_{j'}$ and $\mathcal{P}_{j''}$ are the two "pasts" upstream of $\mathcal{P}$.

Alternatively, following the hint, we may use the special rule for zero-sum nodes to obtain the Hadamard transform of $(p(0 \mid \mathbf{r}_{|\mathcal{P}}), p(1 \mid \mathbf{r}_{|\mathcal{P}}))$ simply by componentwise multiplication of the Hadamard transforms of $(p(0 \mid \mathbf{r}_{|\mathcal{P}_{j'}}), p(1 \mid \mathbf{r}_{|\mathcal{P}_{j'}}))$ and $(p(0 \mid \mathbf{r}_{|\mathcal{P}_{j''}}), p(1 \mid \mathbf{r}_{|\mathcal{P}_{j''}}))$.

By either method, we obtain the following values for the forward messages $\alpha_k = \{\alpha_{0k}, \alpha_{1k}\}$ and their Hadamard transforms $\{\alpha_{0k} + \alpha_{1k} = 1, \alpha_{0k} - \alpha_{1k} = A_k\}$, the backward messages $\beta_k = \{\beta_{0k}, \beta_{1k}\}$ and their Hadamard transforms $\{1, B_k\}$, and the extrinsic information messages $\varepsilon_k = \{\varepsilon_{0k}, \varepsilon_{1k}\}$ and their Hadamard transforms $\{1, E_k\}$.

α_k	α_{0k}	α_{1k}	A_k	
$\alpha_1 = \iota_0$	0.55	0.45	1	$+0.10$
α_2	0.46	0.54	1	-0.08
α_3	0.525	0.475	1	$+0.05$
α_4	0.515	0.485	1	$+0.03$
α_5	0.51	0.49	1	$+0.02$
α_6	0.5035	0.4965	1	$+0.007$
α_7	0.4975	0.5025	1	-0.005

β_k	β_{0k}	β_{1k}	B_k	
$\beta_7 = \iota_7$	0.73	0.27	1	$+0.46$
β_6	0.335	0.665	1	-0.33
β_5	0.45	0.55	1	-0.10
β_4	0.46	0.54	1	-0.08
β_3	0.475	0.525	1	-0.05
β_2	0.515	0.485	1	$+0.03$
β_1	0.49	0.51	1	-0.02

ε_k	ε_{0k}	ε_{1k}	E_k	
$\varepsilon_0 = \beta_1$	0.49	0.51	1	-0.02
ε_1	0.5015	0.4985	1	$+0.003$
ε_2	0.5015	0.4985	1	$+0.003$
ε_3	0.498	0.502	1	-0.004
ε_4	0.4985	0.5015	1	-0.003
ε_5	0.4965	0.5035	1	-0.007
ε_6	0.5015	0.4985	1	$+0.003$
$\varepsilon_7 = \alpha_7$	0.4975	0.5025	1	-0.005

Notice how the reliability of the forward and backward APP messages α_k and β_k degenerates as more and more intrinsic information messages ι_k are incorporated into them.

The APP vectors $\{p(Y_k = 0 \mid \mathbf{r}), p(Y_k = 1 \mid \mathbf{r})\}$ for the symbol variables Y_k are ultimately obtained by componentwise multiplication of ι_k and ε_k, normalized. We note that for all k, since $\varepsilon_{0k} \approx \varepsilon_{1k} \approx \frac{1}{2}$, we have $\{p(Y_k = 0 \mid \mathbf{r}), p(Y_k = 1 \mid \mathbf{r})\} \approx \{\iota_{0k}, \iota_{1k}\}$; *i.e.,* the intrinsic information ι_k dominates. Thus if hard decisions are made on each symbol, the result is the same as if hard decisions had been made symbol-by-symbol based solely on the channel outputs r_k, and the resulting sequence of hard decisions is not a code sequence in the $(8, 7, 2)$ code.

In contrast, suppose that we perform the max-product (equivalent to the min-sum) algorithm with this received sequence. We obtain the following values:

α_k	α_{0k}	α_{1k}
$\alpha_1 = \iota_0$	0.55	0.45
α_2	0.45	0.55
α_3	0.55	0.45
α_4	0.55	0.45
α_5	0.55	0.45
α_6	0.55	0.45
α_7	0.45	0.55

β_k	β_{0k}	β_{1k}
$\beta_7 = \iota_7$	0.73	0.27
β_6	0.27	0.73
β_5	0.35	0.65
β_4	0.35	0.65
β_3	0.35	0.65
β_2	0.65	0.35
β_1	0.35	0.65

ε_k	ε_{0k}	ε_{1k}
$\varepsilon_0 = \beta_1$	0.35	0.65
ε_1	0.55	0.45
ε_2	0.55	0.45
ε_3	0.45	0.55
ε_4	0.45	0.55
ε_5	0.45	0.55
ε_6	0.55	0.45
$\varepsilon_7 = \alpha_7$	0.45	0.55

In this case, the reliability of a forward or backward message is the minimum reliability of any intrinsic information that is incorporated into it. Eventually, this means that the extrinsic information ε_0 dominates the intrinsic information ι_0 for the least reliable symbol Y_0, so the original "hard decision" is "corrected" in this case. The same "correction" would be performed by the Viterbi algorithm or by Wagner decoding.

The max-product (min-sum) algorithm finds the maximum-likelihood code sequence, whereas the BCJR (sum-product) algorithm computes the APP vector of each bit. A bit decision based on the maximum APP minimizes the bit error probability, so the bit error probability could be (slightly) lower with the BCJR algorithm. ML sequence detection minimizes the probability of decoding to the wrong codeword. The sequence of maximum-APP bits from the BCJR algorithm may not be a codeword, as we have just seen.

Compare the complexity of BCJR decoding to Viterbi algorithm decoding (Problem 6.6).

The BCJR algorithm is considerably more complex. For each trellis segment, it has to compute the sum-product update rule three times. The straightforward sum-product update rule involves four multiplications and two additions; the simplified rule requires a single multiplication. By contrast, the VA requires four additions and two comparisons per trellis segment. Computation of the intrinsic APP vector is also more complex, and requires estimation of the noise variance σ^2, which could be a problem. Finally, the logic of the BCJR algorithm is more complex, since it is two-way rather than one-way.

In short, the VA is significantly simpler than the BCJR algorithm. For this reason, the VA was preferred for trellis decoding for many years. The BCJR algorithm was resurrected with the advent of turbo codes, where the fact that it produces "soft" (APP) outputs is essential to its role as a part of iterative decoding algorithms.

(For this scenario, Wagner decoding is even simpler; see the solution to Problem 6.6.)