– and there are edges only between nodes in different classes. The graph and the code's parity-check matrix (1.30) are simply related to each other: each parity-check node corresponds to a row of $\mathbf{H}$ and each bit node corresponds to a column of $\mathbf{H}$; for every 1 in $\mathbf{H}$, there is an edge between the corresponding pair of nodes.

Having noticed this connection between linear codes and graphs, one way to invent linear codes is simply to think of a bipartite graph. For example, a pretty bipartite graph can be obtained from a dodecahedron by calling the vertices of the dodecahedron the parity-check nodes, and putting a transmitted bit on each edge in the dodecahedron. This construction defines a parity-check matrix in which every column has weight 2 and every row has weight 3. [The weight of a binary vector is the number of 1s it contains.]

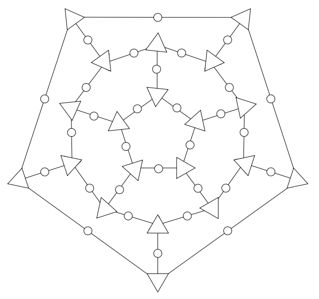
Figure 1.21. The graph defining the $(30, 11)$ dodecahedron code. The circles are the 30 transmitted bits and the triangles are the 20 parity checks. One parity check is redundant.

This code has $N = 30$ bits, and it appears to have $M_{\text{apparent}} = 20$ parity-check constraints. Actually, there are only $M = 19$ *independent* constraints; the 20th constraint is redundant (that is, if 19 constraints are satisfied, then the 20th is automatically satisfied); so the number of source bits is $K = N - M = 11$. The code is a $(30, 11)$ code.

It is hard to find a decoding algorithm for this code, but we can estimate its probability of error by finding its lowest-weight codewords. If we flip all the bits surrounding one face of the original dodecahedron, then all the parity checks will be satisfied; so the code has 12 codewords of weight 5, one for each face. Since the lowest-weight codewords have weight 5, we say that the code has distance $d = 5$; the $(7, 4)$ Hamming code had distance 3 and could correct all single bit-flip errors. A code with distance 5 can correct all double bit-flip errors, but there are some triple bit-flip errors that it cannot correct. So the error probability of this code, assuming a binary symmetric channel, will be dominated, at least for low noise levels f, by a term of order f^3, perhaps something like

$$12\binom{5}{3} f^3(1-f)^{27}. \tag{1.55}$$

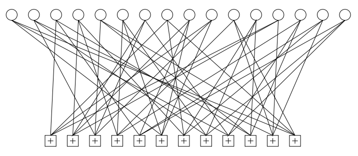
Figure 1.22. Graph of a rate-$1/4$ low-density parity-check code (Gallager code) with blocklength $N = 16$, and $M = 12$ parity-check constraints. Each white circle represents a transmitted bit. Each bit participates in $j = 3$ constraints, represented by $\boxplus$ squares. The edges between nodes were placed at random. (See Chapter 47 for more.)

Of course, there is no obligation to make codes whose graphs can be represented on a plane, as this one can; the best linear codes, which have simple graphical descriptions, have graphs that are more tangled, as illustrated by the tiny $(16, 4)$ code of figure 1.22.

Furthermore, there is no reason for sticking to linear codes; indeed some nonlinear codes – codes whose codewords cannot be defined by a linear equation like $\mathbf{Ht} = \mathbf{0}$ – have very good properties. But the encoding and decoding of a nonlinear code are even trickier tasks.

Solution to exercise 1.10 (p.14). First let's assume we are making a linear code and decoding it with syndrome decoding. If there are N transmitted bits, then the number of possible error patterns of weight up to two is

$$\binom{N}{2} + \binom{N}{1} + \binom{N}{0}. \tag{1.56}$$

For $N = 14$, that's $91 + 14 + 1 = 106$ patterns. Now, every distinguishable error pattern must give rise to a distinct syndrome; and the syndrome is a list of M bits, so the maximum possible number of syndromes is 2^M. For a $(14, 8)$ code, $M = 6$, so there are at most $2^6 = 64$ syndromes. The number of possible error patterns of weight up to two, 106, is bigger than the number of syndromes, 64, so we can immediately rule out the possibility that there is a $(14, 8)$ code that is 2-error-correcting.