Algorithms in the Real World
Assignment #6 Due Date: 12 December 00

Problem 1: (10pt) Recall the bipartite-graph decoding algorithm for Tornado codes (assuming that all check bits are present): find a check bit for which only one neighboring message bit is missing. Calculate the value of that message bit by taking the exclusive-or of the check bit and its other neighboring message bits. Repeat. Prove that if it is possible to recover all of the message bits using this algorithm, then the order in which they are recovered does not matter. I.e., if the algorithm succeeds for one order, then it must succeed for any order.

Problem 2: (10pt) Given the answer to the previous problem, why was it important in the analysis that a degree-one check bit was chosen at random in each step? Prove that by choosing a degree-one check-bit at random, the process has the desired property.

Problem 3: (20pt) It is not difficult to construct a bipartite Tornado code in which all of the message bits can be recovered provided that at most one third are missing (and all check bits are present). For example, it is easy to see how to do this with three message bits and two check bits. Explicitly construct the largest bipartite Tornado code for which you can prove that all message bits can be recovered given that at most one third are missing and all check bits are present. You should attempt to also minimize the number of check bits. (The problem is trivial if the number of check bits is equal to the number of message bits.)

Problem 4: (10pt) The converse of the multicast operation is the accumulate or gather operation. Here a group of senders all want to transmit messages to one receiver. When these messages meet in the network, they can be combined according to some simple rule. For example, two messages might be combined to form a single message by taking the exclusive-or of their data. Compare the difficulty of implementing the accumulate operation on the Internet to that of multicast. What makes this problem easier? What makes it harder?

Problem 5: (10pt) Prove the Eckart and Young theorem mentioned in the last paragraph on page 348 of the paper "Matrices, Vector Spaces and Information Retrieval". Note that this should be a relatively short proof and easy to generate from just the results in that paper. How might one use the theorem to help select k?

Problem 6: (20pt) Consider a term-document matrix A, and assume the columns are all normalized. If we consider the matrix $A^T A$, it gives a similarity matrix on documents based on the cosine measure. Consider the undirected weighted-graph corresponding to this matrix (it is undirected since the matrix is symmetric). Now consider running a graph partitioning algorithm that recursively divides the graph in two equal sized parts by trying to minimize the sum of the edge-weights that are cut. This generates a separator tree.

(A) Explain how you might use this tree to better compress the inverted index of A.

(B) Finding the lowest-weight cut that evenly partitions a graph is NP-hard. Can you suggest a heuristic based on Latent Semantic Indexing for making the first cut?