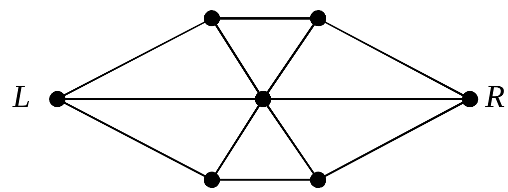
CMPE 177
Applied Graph theory and Algorithms
Summer 2009

Some Remarks on Strategy for Shannon's Switching Game

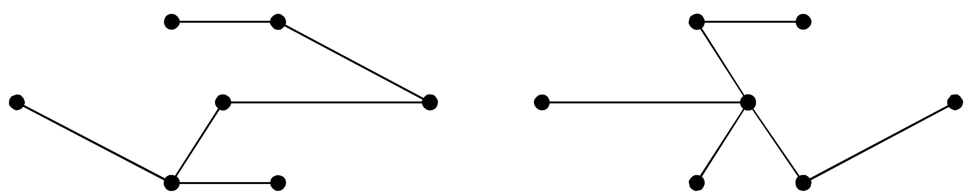
A *spanning tree* in a connected graph *G* is simply a spanning subgraph which is also a tree. In other words it is a subtree of *G* which includes all vertices of *G*. In the context of Shannon's switching game we will say *tree* to mean a spanning tree in the network. Necessarily any tree contains exactly one path from *L* to *R*. If any edge *on that path* is removed, the resulting subgraph is called a *near-tree*.

Examples

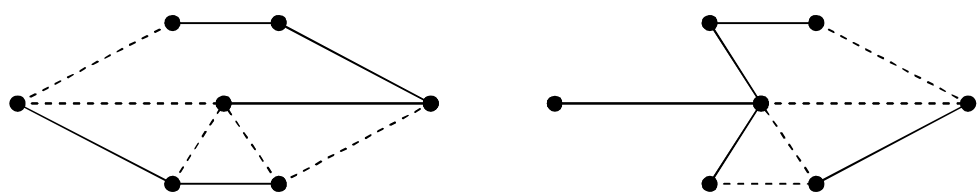
A network *G*:



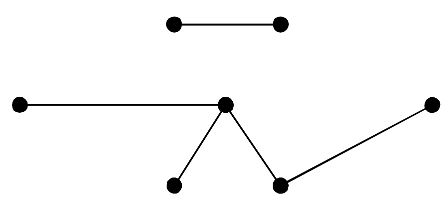
Trees in *G*:



Near-trees in *G*:



Neither a tree nor a near tree:



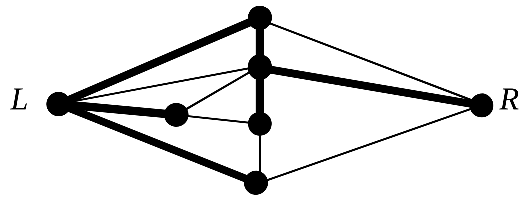
Note that in General there is more than one way to augment a near-tree to a tree, as shown by the dashed lines above. Every near-tree is a forest consisting of two components, each containing one terminal of the network. The set of edges which when added to a near-tree form a tree, themselves form a *cutset*, i.e. a minimal set of edges separating *L* from *R*.

We will consider networks in which the edge set can be partitioned into:

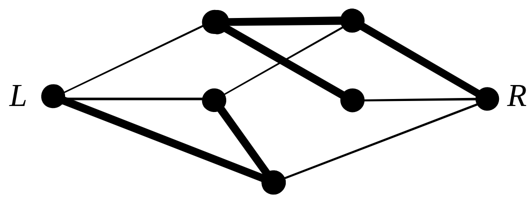
(a) Two trees,
(b) Two near trees
(c) A tree and a near-tree

Examples

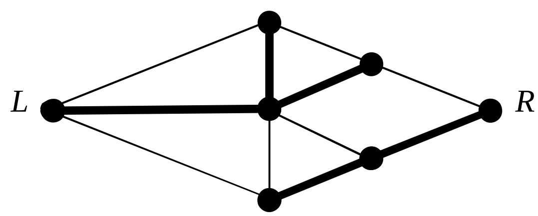
A network of type (a):



A network of type (b):



A network of type (c):

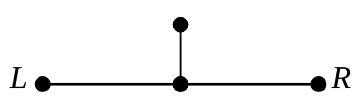


Recall that every network can be classified as one of the following types:

(1) Favorable to C,
(2) Favorable to D, or
(3) Favorable to the first player

Theorem
(a)$\Rightarrow$(1), (b)$\Rightarrow$(2), and (c)$\Rightarrow$(3).

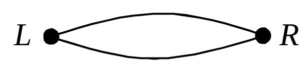
Remarks
It is important to note that the converses to the above assertions are false. For instance, the following network is obviously favorable to D, and yet it cannot be partitioned into two edge-disjoint near-trees.



Let n denote the number of vertices in a network, and m the number of edges. The treeness theorem tells us that (a)$\Rightarrow m = 2n-2$, (b) $\Rightarrow m = 2n-4$, and (c) $\Rightarrow m = 2n-3$. Therefore categories (a), (b), and (c) are mutually exclusive. By the same token, types (a), (b), and (c) do not exhaust all possible networks, so there are networks to which the theorem does not apply. However, any network which can be obtained from one of type (a) by adding edges must certainly be favorable to C, since C can just play as if the extra edges are not present. Similarly any network which can be obtained from one of type (b) by deleting edges is favorable to D.

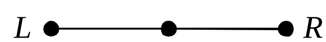
Proof:
Throughout we will interpret C's move as a contraction rather than a reinforcement, as described in the project handout.

(a)$\Rightarrow$(1)
Let T_1 and T_2 be two edge-disjoint spanning trees in *G*. Assume with no loss of generality that D plays first, and that an edge from T_1 is destroyed. The removal of that edge disconnects T_1 into two components. Since the tree T_2 is still intact, it contains an edge which joins vertices from those two components. If C now responds by contracting such an edge, the resulting network will still be of type (a) since T_1 will have been reconnected. Suppose C continues to respond in the fashion, i.e. contracting edges in the tree opposite to the one in which D destroys an edge in such a way as to reconnect the disconnected tree. Then after each round of play the network will still be of type (a). But after each round there is one less vertex and two less edges. After $n-2$ rounds there are 2 vertices and $2\cdot 2-2=2$ edges left. The only such network of type (a) is:



At this point we see that C cannot lose. No matter which edge D destroys, C contracts the other, winning the game. Thus C has a forced win in the original network.

(b)$\Rightarrow$(2)
Suppose *G* consists of two edge-disjoint near-trees H_1 and H_2. Assume without loss of generality that C plays first and an edge *e* of H_1 is contracted. There are two cases to consider. Either (i) *e* joins two vertices in the same component of H_2, or (ii) *e* joins vertices in different components of H_2. In case(i) a unique cycle will be formed in H_2. If D now destroys any edge on that cycle, the resulting network will again be of type (b). This is because H_1 has not been disturbed, and the remaining edges of H_2 are again acyclic. In case(ii) a path from *L* to *R* will have been created in H_2. If D now destroys any edge on that path, the network will be of type (b), since again H_1 has not changed and the surviving edges of H_2 will form a near-tree. Continuing in this way we reach, after $n-3$ rounds of play, a point where there are 3 vertices and $2\cdot 3-4=2$ edges. The only such network of type (b) is:



which is a winning position for D. Therefore D had an advantage in the original network.

(c)$\Rightarrow$(3)
Now suppose the edges of *G* partition into a tree *T* and a near-tree *H*. If C moves first by contracting an edge of *T* which joins the two components of *H*, then the resulting network will be of type (a), and therefore favorable to C. If D moves first by removing an edge from the unique path in *T* which joins *L* to *R*, then the resulting network will be of type (b), which favors D. Therefore the player who moves first has a forced win in this network. ///

While the above theorem is obviously of some importance in building a strategy for Shannon's Switching game, it's full implications are not clear, since there is no guarantee that the network on which your program will be required to play will be one of type (a), (b), or (c). Even if such a guarantee were in effect, finding a decomposition of the network into subgraphs of the required kinds could consume much computing time, since the search space may be quite large. One possible heuristic might be to spend a limited amount of time locating two disjoint spanning trees (say), and if that fails, move on to some other strategy. In any case the preceding theorem is presented in order to stimulate your thinking, not to limit it.

Exercise Classify the following networks as either (a), (b), or (c).