Scientific Applications
C.L. Lawson Editor

An Implemented Graph Algorithm for Winning Shannon Switching Games

Stephen M. Chase
IBM Thomas J. Watson Research Center*

In this tutorial paper a computer program which wins Shannon Switching Games is described. Since these games are played on graphs, the program is a good example of the implementation of graph algorithms.

The two players in a Shannon Switching Game, CONNECT and CUT, have nonsimilar goals. Either CONNECT, CUT, or the player moving first is guaranteed the existence of a winning strategy. The simple strategy explained in this paper is valid in all three cases. In fact, the major routines never need to know whether the computer is CONNECT or CUT.

Key Words and Phrases: graph algorithms, graph processing, Shannon Switching Games, game playing, graph theory, positional games, demonstration programs, game theory, spinning trees

CR Categories: 3.69, 5.32

* P.O. Box 218, Yorktown Heights, NY 10598. This work was done while the author was at the University of Illinois, Department of Computer Science, Urbana, Illinois.

Introduction

A computer program which wins Shannon Switching Games [1, 2, 4] is described in this paper. Since these games are played on graphs, the program serves as a good example of the implementation of graph algorithms. The ease of communication between a human player and the computer through an interactive graphics console and the simplicity of the rules of the game make this program an excellent computer demonstration as well.

The paper begins with a description of the game. Useful concepts are then defined. The algorithm is presented, followed by an example of its execution. Finally, a few implementation details are given.

How the Game Is Played

The definition of the game and a description of how the demonstration looks to the user are best given together as follows. By pointing a light pen to the screen of a DEC 338 display console (the program runs on an attached PDP8), the user first defines his own game by drawing a graph. (See Figure 1.) Twelve nodes are preset on the screen, including two distinguished nodes on the left and right sides of the screen. By pointing to these nodes, the user specifies which branches the computer should draw in. He may also move nodes around the screen to give the graph any shape he wishes (incident branches follow automatically). After he has given the graph, the game is played on the branches. There are two players who move alternately. A move consists of claiming a branch which has not yet been claimed. The CONNECT player wins if a path between the two distinguished nodes is contained in his set of branches. Otherwise, the CUT player wins (i.e. if every path be-

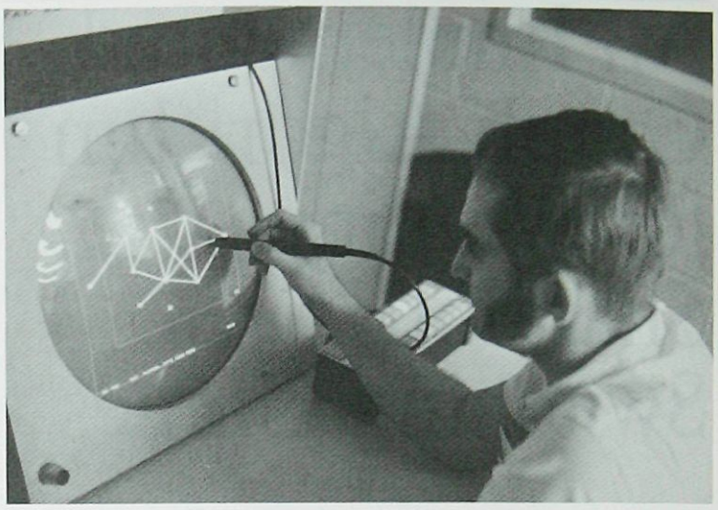
tween the two distinguished nodes contains a branch claimed by CUT). On the screen, CUT moves vanish and CONNECT moves blink.

Games fall into three categories depending on the structure of the graph defined by the user: either (1) CONNECT or (2) CUT has a winning strategy regardless of who moves first, or else (3) whoever moves first has a winning strategy. The computer replies with one of the following messages:

1. I AM THE CONNECT PLAYER, YOU MOVE FIRST.
2. I AM THE CUT PLAYER, YOU MOVE FIRST.
3. SEE MY FIRST MOVE [a branch blinks on the screen], YOU CHOOSE WHICH PLAYER YOU WANT TO BE.

The user indicates his moves by pointing the light-pen to the branch he wants to claim. His move and the computer's reply are instantly modified on the screen. The computer is allowed to pass its turn; in this game, it never hurts to have to move. The computer wins every game (it is surprising how quickly this discourages people from playing many games).

Fig. 1



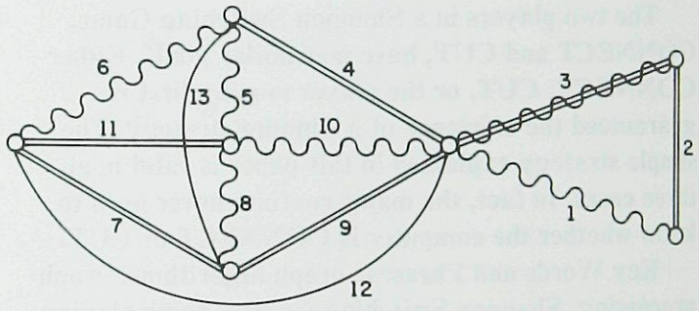
Useful Concepts

The cardinality of a set S is denoted by $|S|$. The symmetric difference (exclusive or) of sets S_1 and S_2 is denoted by $S_1 \oplus S_2$ (the set $S_1 \oplus S_2 \oplus \cdots \oplus S_k$ contains only those elements which belong to an odd number of the sets $S_1, S_2, \cdots, S_k$).

Given a sequence $\langle n_0, b_1, n_1, b_2, \cdots, n_{k-1}, b_k, n_k \rangle$, where the endpoints of branch b_i are the nodes n_{i-1} and n_i, the set of branches $\{b_1, b_2, \cdots, b_k\}$ is called a *path* between n_0 and n_k. The graph G is connected if there is a path between every pair of nodes. By the nature of Shannon Switching Games, we need only consider connected graphs.

A SPANNING TREE (or simply tree, since every tree considered here will be spanning) is a set T of branches such that between any pair of nodes there is exactly one path contained in T. If the two nodes are the endpoints of a branch b in G, then the path may be denoted by $P(b, T)$.

Fig. 2



THEOREM 1. *For any two spanning trees T_1, T_2 and for any branch $h \in T_1$ there exist an odd number (≥ 1) of branches m such that $m \in P(h, T_2)$ and $h \in P(m, T_1)$.*

PROOF. Let $P(h, T_2) = \{b_1, b_2, \cdots, b_k\}$. Consider the set

$$S = P(b_1, T_1) \oplus P(b_2, T_1) \oplus \cdots \oplus P(b_k, T_1).$$

If P_1 is a path between x and y, and p_2 is a path between y and z, then $p_1 \oplus p_2$ is a path between x and z. Then by induction, S is a path between the endpoints of h. Since $S \subseteq T_1$ (which contains only one such path), $S = \{h\}$. From the definition of $\oplus$, there must be an odd number of branches $m \in \{b_1, b_2, \cdots, b_k\} = P(h, T_2)$ for which $h \in P(m, T_1)$. Q.E.D.

In Figure 2, branches in T_1 are denoted by double lines and those in T_2 are denoted by wavy lines. $P(1, T_1) = \{2, 3\}$. Find the three branches m for $h = 7$.

A two-tree system (TTS) consists of two spanning trees T_1, T_2, such that $|T_1 \cap T_2|$ is minimal (over all such pairs of spanning trees in G).

The Algorithm

Step 1.
[Initialize.]
Wait until human defines the graph G.
Find any spanning tree T_1 in G.
$T_2 \leftarrow T_1$.
Order the remaining branches (in G but not in T_1): $\langle e_1, e_2, \cdots, e_j, e_{j+1}, e_{j+2} \rangle$ where e_{j+1} and e_{j+2} are two additional (nonplayable) branches connecting the two distinguished nodes.
$i \leftarrow 1$.

Step 2.
[Try to decrease $| T_1 \cap T_2 |$ by inserting e_i.
Notation: T_k means T_m where $m \in \{1, 2\}$ and $m = k \pmod 2$; e.g. $T_6 \leftarrow T_7$ means $T_2 \leftarrow T_1$.
B is an array of $N = 12$ sets; each $B(k)$ is a subset of T_k; $B(-1)$ is always empty.]
$b_0 \leftarrow e_i$.
$B(0) \leftarrow \{b_0\}$.
$k \leftarrow 1$.
[*Theorem 2. It is sufficient to consider only those cases where e_i is placed in T_1.* Proof is easy but too long.]

Step 3.
[Fine replaceable branches.]
$B(k) \leftarrow \bigcup_{b \in B(k-1)} P(b, T_k)$.
[$B(k)$ is the set of all possible branches in T_k which can be replaced in T_k by a branch in $B(k-1)$.]
If $B(k) = B(k-2)$ then go to Step 5.
[*Theorem 3. e_i will not help decrease $| T_1 \cap T_2 |$.*]
Otherwise, $S \leftarrow (B(k) - B(k-2)) \cap T_{k+1}$.
If $S \neq \phi$ then pick b_k from S and go to Step 4 [because $b_k \in T_1 \cap T_2$, e_i can help decrease $| T_1 \cap T_2 |$].
Otherwise, $k \leftarrow k + 1$.
[*Theorem 4. e_i cannot be placed in T_1, decreasing $| T_1 \cap T_2 |$, with k (or fewer) replacements*] [*Theorem 5. $k < N \equiv$ the number of nodes in the graph.* Unfortunately, there are cases where k approaches this limit (for any N)]
Go to Step 3.
[Iterate.]

Step 4.
[Make replacements. *Theorem 6. Since the initial k from Step 3 is minimal, the following exchanges will leave both T_1 and T_2 spanning trees.*]
Pick $b_{k-1} \in B(k-1)$ such that $b_k \in P(b_{k-1}, T_k)$.
[There exists such that a b_{k-1} by definition of $B(k)$.]
$T_k \leftarrow T_k \oplus \{b_{k-1}, b_k\}$
[Replace branch b_k by b_{k-1} in T_k; now if $k > 1$, $b_{k-1} \in T_1 \cap T_2$; if $k = 1$, $b_0 = e_i \in T_1$, and $| T_1 \cap T_2 |$ has decreased by 1]
$k \leftarrow k - 1$.
If $k > 0$ then go to Step 4.
[Continue replacements.]
Otherwise, continue.

Step 5.
[Iterate Step 2.]
If $i < j + 2$ and $T_1 \cap T_2 \neq \phi$ then $i \leftarrow i + 1$ [there is hope for decreasing $| T_1 \cap T_2 |$]; go to Step 2.
Otherwise, continue.

Step 6.
[Evaluate game.]
If $e_{j+1} \in T_2$ [equivalently: if $e_{j+2} \in T_1$], then choose CUT.
If $e_{j+1} \in T_1$, then choose to move first; $h \leftarrow e_{j+1}$ [regard the unplayable e_{j+1} as human's first move; now find the appropriate reply; call it the machines first move regardless of who is CONNECT or CUT]; go to Step 8.
Otherwise, choose CONNECT.
[*Theorem 7. The above decisions correctly evaluate the game.*]

Step 7.
[Get human move.]
Wait for human move, call it h. If $h \notin T_1 \oplus T_2$, then go to Step 7 [machine passes].
Otherwise, continue.

Step 8.
[Find reply.]
Let T be the tree containing h, T' the tree not containing h. Find a move m such that $m \in P(h, T')$ and $h \in P(m, T)$.
[Such a branch m exists by Theorem 1. Note that this step does not need to know who is CONNECT or CUT.]

Step 9.
[Update.]
Let T be the tree containing CUT's move.
$T \leftarrow T \oplus \{$CUT's move, CONNECT's move$\}$.
[For example, in Figure 2, if CUT's = 5 and CONNECT's = 11, then T_1 remains unchanged, but $T_2 \leftarrow \{1, 2, 6, 8, 10, 11\}$.]

Step 10.
[Game over?]
Make a simple test to see if either player has won.
If machine has won, then ring bell.
If human resigns, then go to step 1.
Otherwise, go to Step 7.

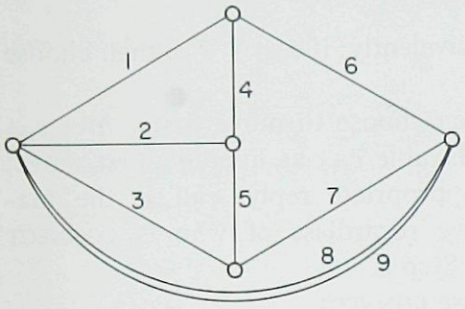
After all moves have been made, T_1 will be a subset of CONNECT's moves (due to Step 9). Thus CONNECT's moves contain a path between the distinguished nodes. If the computer is CONNECT, this path is O.K., but if the human is CONNECT, this path consists of a nonplayable branch (e_{j+1} or e_{j+2}). In either case, the computer wins.

Because this single algorithm contains both CONNECT's and CUT's strategy, it is more efficient than similar algorithms (such as [1]).

Example

Consider the graph in Figure 3. Branches 8 and 9 are the unplayable branches connecting the two distinguished nodes (see Step 1).

Fig. 3



Step	Action
1	$T_2 \leftarrow T_1 \leftarrow \{1, 2, 5, 6\}$; $i \leftarrow 1$.
2	$B(0) \leftarrow \{b_0 \leftarrow e_i = 3\}$; $k \leftarrow 1$.
3	$B(1) \leftarrow \{2, 5\}$; $S \leftarrow \{2, 5\}$; $b_1 \leftarrow 2$.
4	$T_1 \leftarrow \{1, 3, 5, 6\}$; $k \leftarrow 0$.
5	$i \leftarrow 2$.
2	$B(0) \leftarrow \{b_0 \leftarrow 4\}$ $k \leftarrow 1$.
3	$B(1) \leftarrow \{1, 3, 5\}$; $S \leftarrow \{1, 3, 5\}$; $b_1 \leftarrow 1$.
4	$T_1 \leftarrow \{3, 4, 5, 6\}$; $k \leftarrow 0$.
5	$i \leftarrow 3$.
2	$B(0) \leftarrow \{b_0 \leftarrow 7\}$; $k \leftarrow 1$.
3	$B(1) \leftarrow \{4, 5, 6\}$; $S \leftarrow \{5, 6\}$; $b_1 \leftarrow 6$.
4	$T_1 \leftarrow \{3, 4, 5, 7\}$; $k \leftarrow 0$.
5	$i \leftarrow 4$.
2	$B(0) \leftarrow \{b_0 \leftarrow 8\}$; $k \leftarrow 1$.
3	$B(1) \leftarrow \{3, 7\}$; $S \leftarrow \phi$; $k \leftarrow 2$.
3	$B(2) \leftarrow \{1, 2, 5, 6\}$; $S \leftarrow \{5\}$; $b_2 \leftarrow 5$.
4	$b_1 \leftarrow 3$; $T_2 \leftarrow \{1, 2, 3, 6\}$; $k \leftarrow 1$.
4	$T_1 \leftarrow \{4, 5, 7, 8\}$; $k \leftarrow 0$.
5	$T_1 \cap T_2 = \phi$.
6	$8 \notin T_2$; $8 \in T_1$, so choose to move first; $h \leftarrow 8$.
8	$m \leftarrow 1$ [machines first move]. Say human chooses to be CONNECT.
9	CUT's = 1, CONNECT's = 8, so $T_2 \leftarrow \{2, 3, 6, 8\}$.
7	Say human claims branch 7; $h \leftarrow 7$.
8	$m \leftarrow 3$.
9	CUT's = 3, CONNECT's = 7, so $T_2 \leftarrow \{2, 6, 7, 8\}$.
7	Say human claims branch 2; $h \leftarrow 2$.
8	$m \leftarrow 5$.
9	$T_1 \leftarrow \{2, 4, 7, 8\}$.
7	$h \leftarrow 6$, say.
8	$m \leftarrow 4$.
9	$T_1 \leftarrow \{2, 6, 7, 8\}$.
10	CUT wins.

Implementation Details

The game playing program takes up about 3k of the PDP8's 16k memory. The graphics program (modified version of another program) takes another 3k, and display information takes up an amount of storage proportional to the number of branches drawn.

The graph is stored as four arrays indexed by branch number: one array for display information, one for miscellaneous bits of information, and one for each endpoint of the branch.

A tree is stored as an array indexed by node number (0, 1, 2, . . . , 11). The ith entry of this array is the branch pointing to that node i (in this representation, trees are directed arbitrarily).

The output set of the P function is stored in a simple list:

Location	Content
L	N
$L + 1$	branch 1
⋮	⋮
$L + N$	branch N

Connection matrices are used and this accounts for the restriction to 12 nodes (PDP8 words contain 12 bits). All other parts of the program could be extended easily to the general case.

Received August 1969; revised April 1971

References

1. Bruno, J., and Weinberg, L. A constructive graph-theoretic solution of the Shannon Switching Game. *IEEE Trans. Circuit Theory CT-17*, 1 (Feb. 1970), 74–81.
2. Busacker, R., and Saaty, T. *Finite Graphs and Networks: An Introduction with Applications*. McGraw-Hill, New York, 1965, pp. 167–169.
3. Kishi, G., and Kajitani, Y. Maximally distant trees and principal partition of a linear graph. *IEEE CT-16*, 3 (Aug. 1969), 323–330.
4. Lehman, A. A solution to the Shannon Switching Game. *SIAM J. 12*, No. 4 (Dec. 1964), 687–725.