3-Terminal Games
Introduction: The Setup

• Given a directed (cyclic or acyclic) graph containing an origin and 3 terminal vertices.
• The non-terminal vertices are divided among a given number of players.
• Each player has an ordering of preferred terminals for the play to terminate in, where a cycle is the least preferred option.
Introduction: The Play

- Each player must choose exactly one outgoing edge from each vertex he controls.
- The result of the play is whichever Terminal vertex the is reached from by following the edges from the origin. If no Terminal is reached the play has entered a cycle.
- If the edges are chosen such that no player can benefit by switching his edges the play has entered Nash Equilibrium.
Sample Game

Preferences
1: A>B>C
2: B>C>A
3: C>A>B
Sample Nash Equilibrium Play

Preferences
1: A>B>C
2: B>C>A
3: C>A>B
Theorem

• Every 3-Terminal game contains some Nash Equilibrium that does not terminate in a cycle.
Base Case: A one player game

Player 1 prefers:
A > B > C > cycle
The Nash Equilibrium Play

One prefers:
A > B > C > cycle
The Inductive Step

• Assuming a Nash Equilibrium exists for any k-player game, we want to show that a Nash Equilibrium exists for any k+1-player game.

• We preprocess the graph such that it follows a specific set of rules.

• A Nash Equilibrium on the processed graph must remain a Nash Equilibrium on the original graph.
Step 1: Merge vertices with their optimal terminals.

1: A>B>C
2: B>C>A
3: C>A>B
Result: No player can reach his optimal terminal alone.

1: A>B>C
2: B>C>A
3: C>A>B
Step 2: If any vertex can reach only terminals, merge with the its best.

1: A>B>C
2: B>C>A
3: C>A>B
Result: Every vertex can access some non-terminal vertex.

1: A>B>C
2: B>C>A
3: C>A>B
Step 3: Eliminate connected nodes controlled by one player.

1: A>B>C
2: B>C>A
3: C>A>B
Step 4: Merge any nodes with only one outgoing edge.

1: A>B>C
2: B>C>A
3: C>A>B
And repeat until no more adjustments can be made.

1. A > B > C
2. B > C > A
3. C > A > B
Results of preprocessing

• Every node has multiple edges, at least one of which points to a non-terminal.
• No node can reach its best terminal.
• No node points to another node controlled by the same player.
• A Nash Equilibrium in the new graph is a Nash Equilibrium in the original.
Setting N

• We aim to set a random players nodes (let’s call him N) so that we can find a Nash Equilibrium for the remaining k players. We will then convert this to a Nash Equilibrium for all players.

• In the process we will partition the graph.
Stage 1: Create Area B

- For the 2\textsuperscript{nd} highest terminal in player N’s order of preference (hereafter B): If any player N node can reach terminal B or another B-node, set its edge toward that node and label it B.
- Label any node belonging to another player which only has edges to B nodes with a B.
Stage 2: Create Area AB

• For player N highest ranked terminal (call it A): Label all nodes that only reach A and B terminals and/or B and AB nodes with an AB.
• For any player N node that can reach an AB node, set its edge toward that node and label it AB.
• This results in the partitioning of the graph into AB and not AB (or AB’) areas.
Finally set the remaining N nodes

• Using a reverse breadth-first-search (i.e. following edges from destination to origin) from nodes C, B and A in that order we set N’s remaining nodes to point towards terminals.

• This is simply to avoid cutting off access to terminals (and forcing a cycle).
Solve the resulting k player game

• Recall that we’re assuming that all k player, 3 terminal games can be solved, so take the k player graph that results from setting N and find the Nash Equilibrium.

• We want to turn the k player NE into a k+1 player (with N) NE.

• We address the 3 possible cases: The k player equilibrium play points to A, to B or to C (N’s 1\textsuperscript{st}, 2\textsuperscript{nd}, and 3\textsuperscript{rd} best terminals, in order).
Case 1: NE to A

- This is a Nash Equilibrium for players 1 through k already.
- Player N also has no incentive to switch from the edges chosen for him as the play currently points to his optimal terminal.
Case 2: NE to B

• Lemma: We can redirect all edges away from non-B terminals while maintaining NE.
S: The node that was switched from a non-B terminal node. X1 and X2 are 2 nodes controlled by one player P.
S: The node that was switched from a non-B terminal node.
X1 and X2 are 2 nodes controlled by one player P.

P:  \[ T' \] >  \[ B \] >  \[ T \]
X2 should have been merged with T' in the preprocessing!
NE is maintained

• Since none of the non-N players can reach preferred terminals they remain in Nash Equilibrium.
• Since N cannot reach A, he is content to maintain the play to B.
Case 3: NE to C

- Lemma 1: We can redirect all edges away from non-B terminals while maintaining NE.
- PROVEN (as above).
- Lemma 2: We can set the edges of all AB’ nodes away from AB without breaking the NE.
Recall that all non-N AB vertices only had routes to terminals in the partitioning.
Hence:

\[
P: \quad \text{B} > \quad \text{C} > \quad \text{A}
\]
These steps give us this graph:

Area AB’
NE is maintained

• Since none of the non-N players can reach preferred terminals they remain in Nash Equilibrium.

• Since N cannot reach the AB side of the partition using any of his vertices (otherwise they’d be part of AB) or using any other players vertices (all point away), C is the best terminal he can reach and the graph is in proper Nash Equilibrium.
By induction: Every 3-terminal game contains a proper Nash Equilibrium.

• Every 1-player, 3-terminal games contains a non-cyclical Nash Equilibrium
• If every k-player game contains an NE then every k+1-player game contains an NE.
Theorem 2:

We can find NE in polynomial time.

• Simply use our method recursively.
• Since every step in the sorting sets each node at most once, every step can be done in $O(N)$ time.
• Over $P$ players (where $P<N$) we run the process $P$ times, hence finding NE takes $O(N^2)$ time.
Thank you.

Special thanks to my mentor, Dr. Endre Boros, and to the Center for Discrete Mathematics and Computer Science.