The Ninja Assassin Wonderwall Game

Brian Schiller, advised by Dr. Tilmann Glimm
Math Department, Western Washington University, Bellingham, WA

Abstract

In this summer camp game, each of n players chooses one person to be their Ninja Assassin and another to be their Wonderwall. When the game starts, each player runs to try to keep their Wonderwall on the line between themself and their Ninja Assassin. We create an algorithm, based on finding cycles in directed graphs, that creates a configuration satisfying every player's constraints or says that no such configuration exists. We also use a computer model to estimate the probability that a given game has such a configuration.

How to Play

Ninja Assassin Wonderwall is a game played at summer camps with around 10-30 players. In order to play, each player should:

- Choose someone to be their ninja assassin.
- Choose someone to be their wonderwall.
- (don't tell anyone who you picked)
- When I yell "Go", run to keep yourself on the safe side of the line between your ninja assassin and your wonderwall.
- (Stop when dinner is ready)

Funny thing, sometimes the games stop early. The players slow down… and stop. Each player is in a position where their wonderwall protects them from their ninja assassin.

Definitions

A *Ninja Assassin Wonderwall problem* is a set P together with two functions, w: $P \to P$, n: $P \to P$ that satisfy $w(p) \neq p, n(p) \neq p$, and $w(p) \neq n(p)$ for all $p \in P$. (That is, players can't choose themselves and can't choose the same person twice).

A *solution* to a Ninja Assassin Wonderwall problem is a function S: $P \to R^n$ such that:

- S is one-to-one (two players may not occupy the same space)
- For each $p \in P, w(p)$ lies on the line segment between p and $n(p)$.

Problem Directions

- How can we tell if a game has a solution?
- Given a game, construct a solution or say that none exists.
- Try to determine how the probability of a solution existing changes for games of different sizes.

Challenge: which problem has a solution?

p	$w(p)$	$n(p)$
A	C	B
B	C	A
C	B	D
D	C	A

One of these two problems has a solution, a configuration of players that satisfies every player's constraints. The other has no such configuration– those players will keep running forever (or until dinner time).

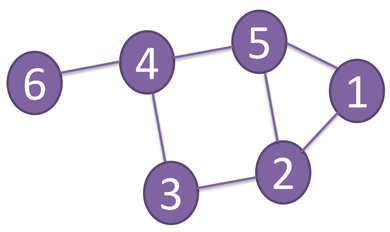
p	$w(p)$	$n(p)$
A	B	C
B	D	A
C	B	A
D	B	A

How can we tell whether a game has a solution?

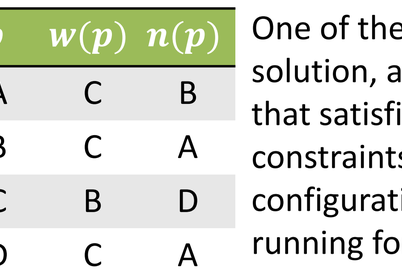
- Idea: Try all the orderings!

How long would this take? With n players, there are $n!$ permutations, so it could take that many steps. This is too slow: if we could check 1000 permutations each second, solving a 20 player game would take $7.71 \cdot 10^7$ years. We need a better idea.

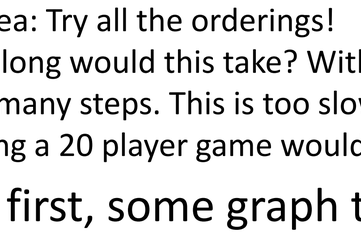
But first, some graph theory terms



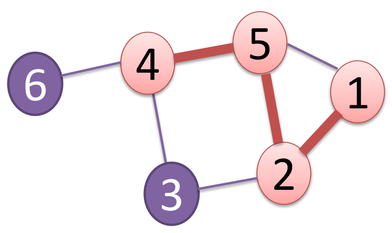
A *graph* is a collection of vertices V together with a collection of edges E that connect pairs of vertices.



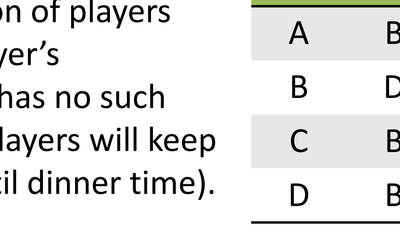
A *directed graph*, or digraph, is a graph whose edges have a direction associated with them.



A *directed acyclic graph*, or DAG, is a digraph with no cycles.



A *path* in a graph is a sequence of vertices in which there is an edge to connect each adjacent pair of vertices.

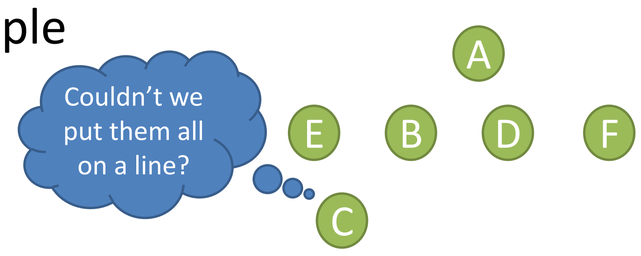


A *cycle* in a digraph is a path which starts and ends at the same vertex.

<6,4,3,2,5,1>

A *topological sort* on a DAG is a linear ordering of its vertices such that each node appears before every node with which it has outbound edges.

Small Example

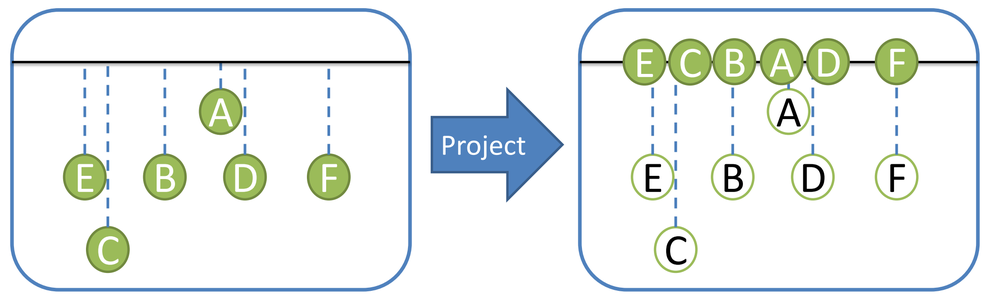


Each row of the table lists a single player's choices: the player, their wonderwall, their ninja assassin. Notice that each player has their wonderwall protecting them from their ninja assassin.

p	$w(p)$	$n(p)$
A	B	C
B	D	F
C	B	A
D	B	E
E	D	F
F	D	B

Fork me on GitHub!

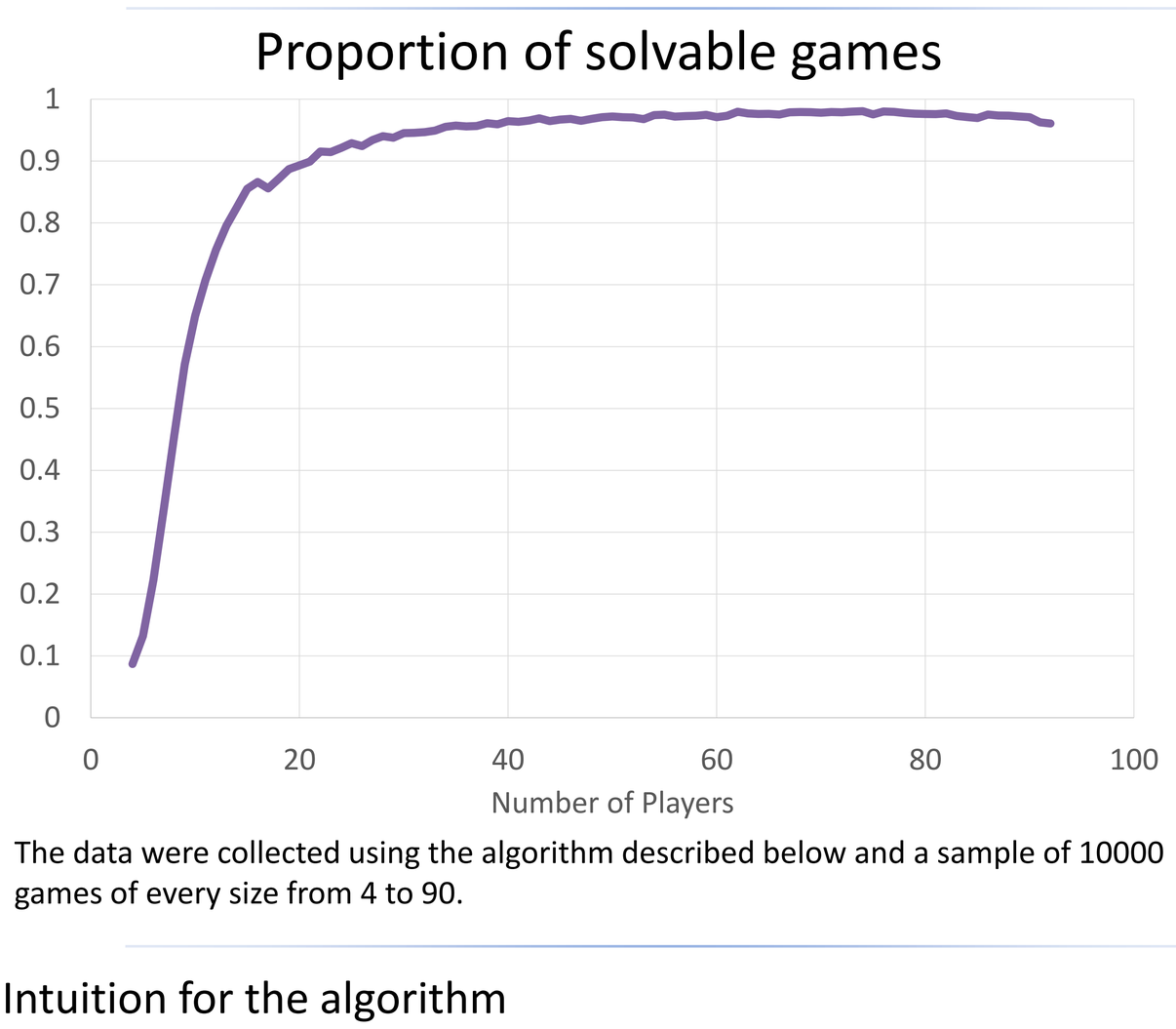
code and thesis at: http://goo.gl/VZnH6



Theorem: *Any Ninja Assassin Wonderwall game which has a solution in R^n has a solution in R.*

Why is this important?

- If we're searching for solutions, we don't have to worry about arrangements in more than one dimension

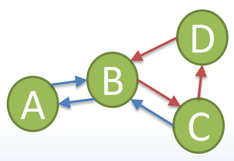


The data were collected using the algorithm described below and a sample of 10000 games of every size from 4 to 90.

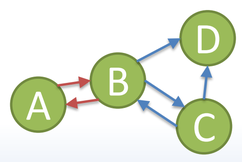
Intuition for the algorithm

1. If a permutation of the players satisfies player p's constraints, then $p, w(p), n(p)$ appear in that order or its reverse.
2. We will build a DAG where a path exists from player p to player q if "p appears to the left of q in a solution to the Ninja Assassin Wonderwall problem".
3. We construct the edge set of our graph by taking either $p \to w(p) \to n(p)$ or the reverse, $p \leftarrow w(p) \leftarrow n(p)$ for each player p.
4. If the resulting graph is acyclic, a topological sort will produce a solution.

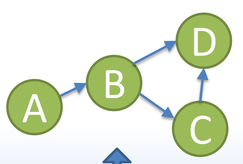
p	$w(p)$	$n(p)$
A →	B →	C
B →	C →	D
C →	B →	A
D →	B →	A



p	$w(p)$	$n(p)$
A →	B →	C
B →	C →	D
C →	B →	A
D ←	B ←	A



p	$w(p)$	$n(p)$
A →	B →	C
B →	C →	D
C ←	B ←	A
D ←	B ←	A



That one is acyclic!

A topological sort gives<A,B,C,D> , which (sure enough) is a solution!