1 Introduction

Game theory is a series of tools and methods which attempt to model rational[1] behavior given agents with payoffs. In a simple game theory model, agents have mutual knowledge[2] and play simultaneous their actions. Depending on all agents actions, they receive a payoff. The agents are thus attempting to maximize their payoffs without knowing what the other player will do. Hence, game theorist are interested in finding nash equilibriums where each player is finds a strategy[3] which optimizes their payoff regardless of what the other player does. This is solved through considering a payoff matrix:

		Player Y	
		A	B
Player X	A	(x_1, y_1)	(x_2, y_2)
	B	(x_3, y_3)	(x_4, y_4)

Here we have two agents (player X, and player Y) with two choices (A and B) which lead to different payoffs for each agent (x, y). Hence, to find the Nash equilibrium(s), we need to find the optimal payoff given each choice.

Algorithm 1 Finding Nash Equilibrium

1: **procedure** SOLVE FOR PLAYER X(agents, choices, payoffs)
2: Assume player Y chooses A, find ordering of x_1, x_3
3: Assume player Y chooses B, find ordering of x_2, x_4
4: **procedure** SOLVE FOR PLAYER Y(agents, choices, payoffs)
5: Assume player X chooses A, find ordering of y_1, y_2
6: Assume player X chooses B, find ordering of y_3, y_4

If there exists (x_i, y_i) which are preferred by both players, this is a Nash Equilibrium.

[1] Complete and transitive, e.g. players can compare all available bundles and rank them with transitive ordering
[2] All agents know their payoffs, all agents know that other agents know their payoffs, ad infinitum
[3] A move or choice in the game

2 Example: Prisoner's dilemma

An example of this is the prisoners dilemma which may be presented as the following payoff matrix.

		Player Y	
		A	B
Player X	A	$(-1,-1)$	$(-10,0)$
	B	$(0,-10)$	$(-5,-5)$

One can indicate with a hat which outcome given the other player's choice is preferred. Player X prefers 0 over -1 when player Y plays A and player X prefers -5 over -10 when player Y plays B. Player Y preferences are identical.

		Player Y	
		A	B
Player X	A	$(-1,-1)$	$(-10,\widehat{0})$
	B	$(\widehat{0},-10)$	$(\widehat{-5},\widehat{-5})$

Hence in a prisoner's dilemma, although the players would be better off choosing (A, A), through rational behavior, they unfortunately choose (B, B) a worst outcome.

3 Current Code

Currently I solve explicitly for the 2×2 case through a function (find nash) within the method (game) which finds the argmaxs. I want two things, how would one change this code such that it's not if-statements and how would then one generalized to any arbitrary set of finite choices for each actor. One way that I've thought about doing this is to continue to brute force it on a matrix of tuples. I want to explore if there are alternatives to that approach.

```
    def find_nash(self):
        """
        Solves for the nash equilibrium
        """

        max_num_payoffs = max(len(l) for l in self.payoffs)
        index = np.nan*np.zeros([self.players, max_num_payoffs], dtype='int')
        for i in range(0, self.players):
            for j in range(0, len(self.payoffs[i])):
                index[i][j] = np.argmax(self.payoffs[i][j])

        nash = np.zeros([self.players, max_num_payoffs], dtype='int')
        if index[0][0] == 0 and index[1][0] == 0:
            nash[0][0] = 1
        if index[0][1] == 0 and index[1][0] == 1:
            nash[0][1] = 1
        if index[0][0] == 1 and index[1][1] == 0:
            nash[1][0] = 1
        if index[0][1] == 1 and index[1][1] == 1:
            nash[1][1] = 1

        self.nash = nash
```

4 Next steps

4.1 Solving for $m \times n$ choices

This is how a generalized payoff matrix would look like.

		Player Y			
		A	B	...	N
Player X	A	(x_1, y_1)	(x_2, y_2)	...	(x_n, y_n)
	B	(x_{n+1}, y_{n+1})	(x_{n+2}, y_{n+2})	...	(x_{2n}, y_{2n})
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	M	(x_{n*m+1}, y_{n*m+1})	(x_{n*m+2}, y_{n*m+2})	...	(x_{n*m+n}, y_{n*m+n})

4.2 Non simultaneous Games

Trees, these games are actually simpler although more computationally annoying.

4.3 N Players... erm probs just 3 players

		Player Y: A	Player Y: B
Player X	A	(x_1, y_1)	(x_2, y_2)
	B	(x_3, y_3)	(x_4, y_4)

		Player Z: A	Player Z: B
Player X	A	(x_1, z_1)	(x_2, z_2)
	B	(x_3, z_3)	(x_4, z_4)

		Player Y: A	Player Y: B
Player Z	A	(z_1, y_1)	(z_2, y_2)
	B	(z_3, y_3)	(z_4, y_4)