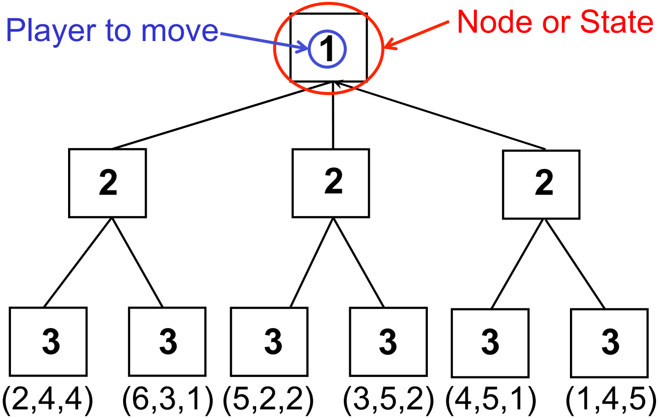
Search Algorithms for Multi-player Games

Presentation by: Brandon Wilson

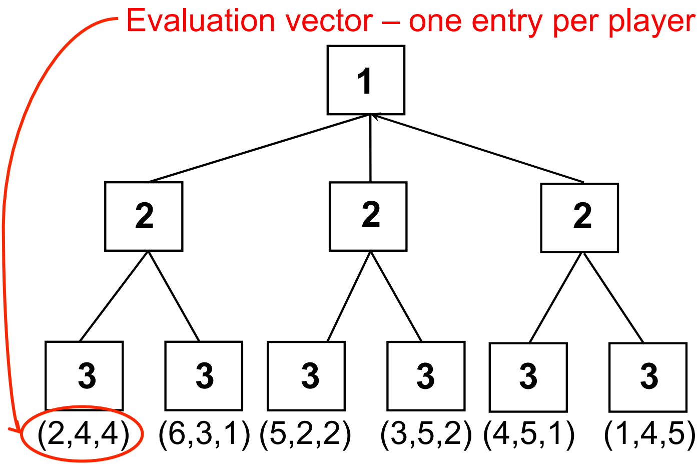
Introduction

- Research in multi-player game-tree search significantly behind work on two-player games
- “Which algorithm to use?” is still open question for multi-player games
- This talk considers games that are:
    - Multi-player (3 or more players)
    - Perfect-information
    - Deterministic

Terminology Related to Game Trees (review)



Terminology Related to Game Trees (review)



Max^n

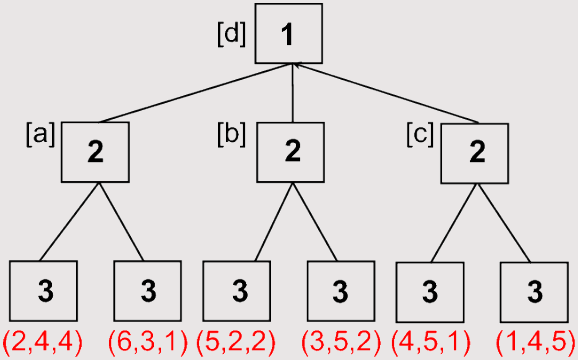
- Claude Shannon's minimax algorithm generalized to n-player games
- Evaluations of leaves are n-tuples, called evaluation vectors
- Max^n values computed assuming players maximize own payoff

Max^n

- Claude Shannon's minimax algorithm generalized to n-player games
- Evaluations of leaves are n-tuples, called evaluation vectors
- Max^n values computed assuming players maximize own payoff

Max^n example

Step 1: Search to limited depth. Apply evaluation function to leaves.

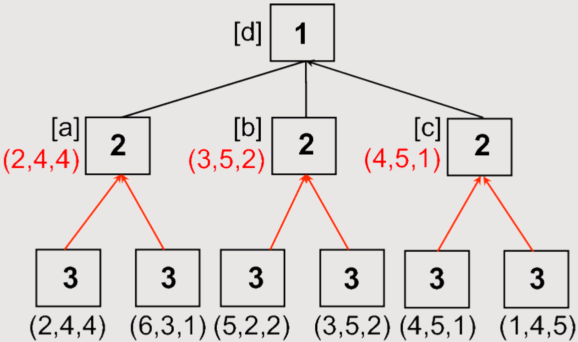


Max^n

- Claude Shannon's minimax algorithm generalized to n-player games
- Evaluations of leaves are n-tuples, called evaluation vectors
- Max^n values computed assuming players maximize own payoff

Max^n example

Step 2: Propagate vectors assuming players maximize own payoff

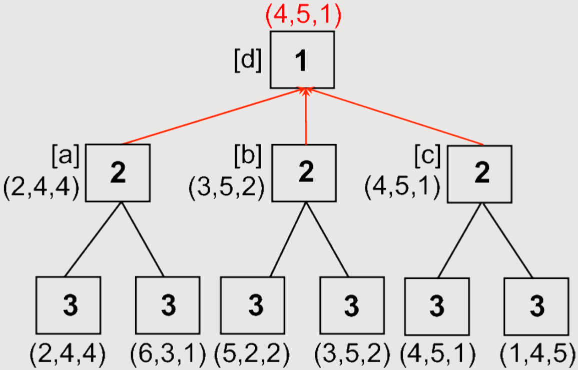


Max^n

- Claude Shannon's minimax algorithm generalized to n-player games
- Evaluations of leaves are n-tuples, called evaluation vectors
- Max^n values computed assuming players maximize own payoff

Max^n example

Step 2: Propagate vectors assuming players maximize own payoff

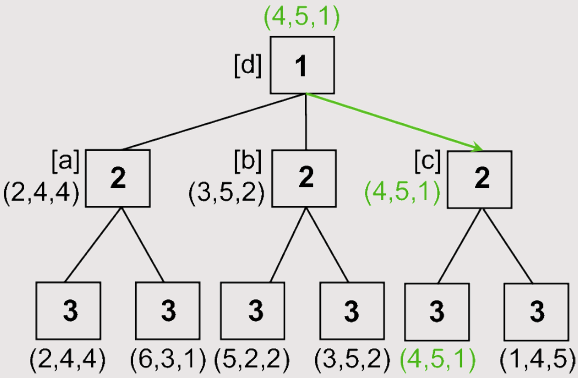


Max^n

- Claude Shannon's minimax algorithm generalized to n-player games
- Evaluations of leaves are n-tuples, called evaluation vectors
- Max^n values computed assuming players maximize own payoff

Max^n example

Step 3: Select move leading to best evaluation vector



Max^n Properties and Limitations

- Many Max^n values may exist for a single tree
  - Value may change drastically depending on tie breaking rule
    E.g., (2,3,3) vs. (2,1,7)
  - Each one is a Nash equilibrium
- Deep pruning (e.g., alpha-beta) is **impossible**
- Shallow pruning (pruning children based on parent bounds) works
  - Requires:
    - Upper and lower bound on each player's score
    - Upper bound on sum of all player's scores
  - Provides minimal benefit

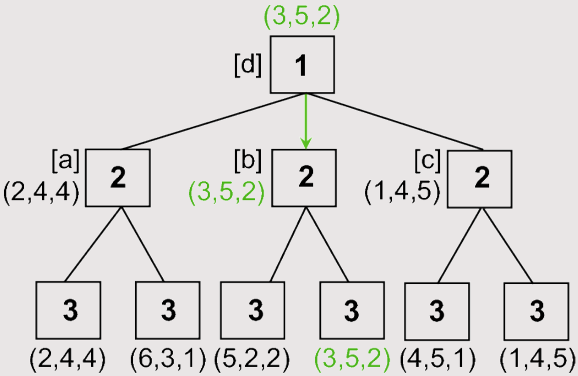
Paranoid

- Assumes all players “gang up” on paranoid player
  - Paranoid player maximizes own payoff on his turn
  - Other players minimize Paranoid’s payoff on their turn

Paranoid

- Assumes all players "gang up" on paranoid player
  - Paranoid player maximizes own payoff on his turn
  - Other players minimize Paranoid's payoff on their turn

Paranoid example



Paranoid Properties and Limitations

- Like Minimax:
  - there is a single paranoid value for a tree
  - the paranoid value is a guaranteed lower bound on one's score
- Alpha-beta can be utilized
  - As number of players increase, benefit of pruning decreases
  - For 3-6 player games, 20-50% deeper search than Max^n
- Paranoid assumption is very pessimistic...
  - May lead to overly cautious play
  - When wrong, deeper search may have negative effect

Motivation

- Algorithms seen thus far experience limited success
- Partially due to the presence of dynamic relationships among players
  - Teams form and dissolve over time
  - Human players may hold grudges
- Social relationships part of strategy in some games (e.g., Risk)
- Max^n and Paranoid make simplifying assumptions:
  - Max^n assumes all players are selfish
  - Paranoid assumes that all opponents attack the Paranoid player

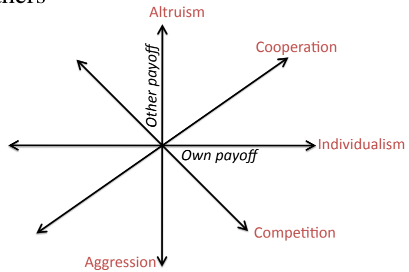
Motivation

- Algorithms seen thus far experience limited success
- Partially due to the presence of dynamic relationships among players
  - Teams form and dissolve over time
  - Human players may hold grudges
- Social relationships part of strategy in some games (e.g., Risk)
- Max^n and Paranoid make simplifying assumptions:
  - Max^n assumes all players are selfish
  - Paranoid assumes that all opponents attack the Paranoid player

Fundamental Question: ***How do we describe and reason about these relationships during gameplay?***

Social Orientations

- *Social Orientation* is the how much one cares about one's own payoff w.r.t. that of others'



Two-player Social Orientation Spectrum

- Social orientations can be represented as a matrix
  - Matrix element (i, j) represents how player i feels about player j's score
  - E.g.

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$$

Player 1 is individualistic

Player 2 is aggressive toward Player 1

Max^n and Paranoid Assumptions as Matrices

Social orientations assumed by Max^n and Paranoid are:

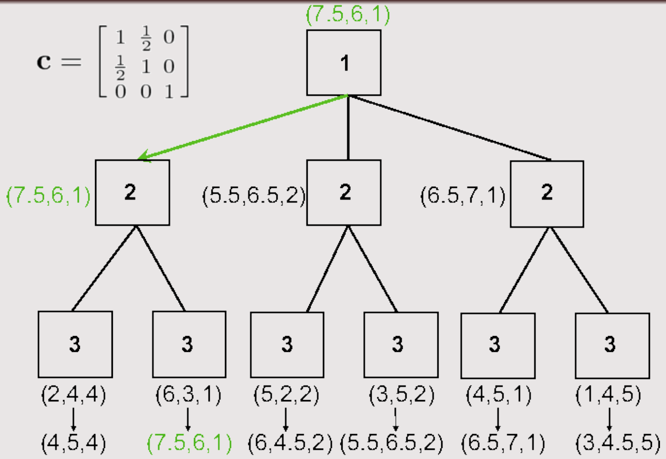
- Max^n can be achieved with the identity matrix
  - For each i, player i cares only about his/her own score (individualist)
  - For example, $c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ for a 3-player game
- Paranoid can be achieved with individualist orientation for paranoid player and aggressive orientations for others
  - For each $i \neq 1$, player i wants to minimize player 1's score
  - For example, $c = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$ for a 3-player game

Socially Oriented Search (SOS)

- *Socially Oriented Evaluation (SOE)* is the dot-product of a social orientation with an evaluation vector
- Assume players maximize SOE during propagation in game tree

Example SOS Search

$$\mathbf{c} = \begin{bmatrix} 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Experimental Domain: Quoridor

- A four-player game played on a 9×9 grid
- Pawn location for each player is the center of one of the four edges
- Each player has 5 walls that can block a path
- Each turn a player can place a wall or move to an adjacent grid location
- Goal is to reach opposite side first

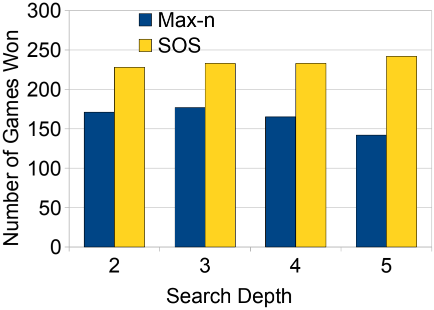
Quoridor board

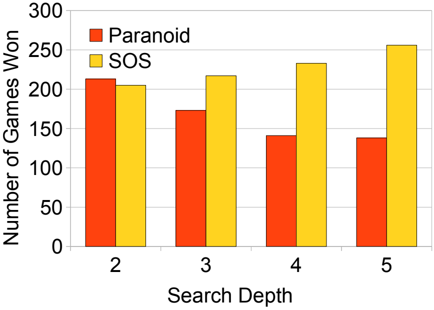
Example mid-game board

Experimental Methodology

- 500 Quoridor games
- Two players with random preferences are generated for each game
- Games played with two sets of players:
    1. $\{\text{Max}^n$, SOS, random_1, random_2$\}$
    2. $\{$Paranoid, SOS, random_1, random_2$\}$

Experimental Results (given true social preferences)





SOS consistently outperformed both Paranoid and Max^n

Ad-hoc Heuristic for Learning Social Preferences

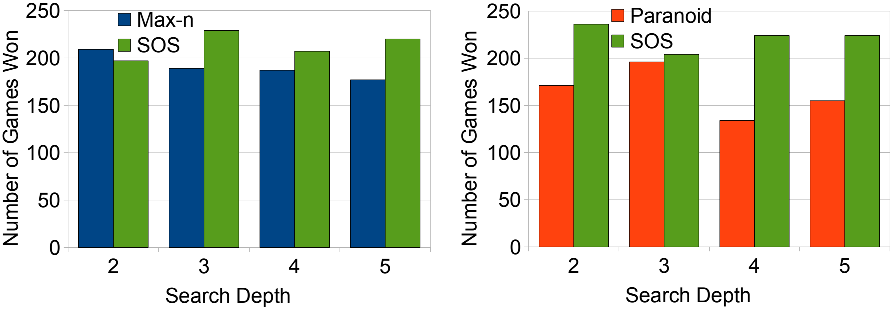
- **Problem:** social orientations are not usually explicitly known
- **Goal:** learn social orientations by observing previous behavior
- Estimate the effect of move from state s_1 to state s_2 as:

$$\Delta(s_1, s_2) = eval(s_2) - eval(s_1)$$

- Estimate player's social orientation as average effect of last k moves.

- Used same experimental setup to test SOS with ad-hoc heuristic

Experimental Results (learned social preferences, $k = 5$)



SOS consistently outperformed both Paranoid and Max^n

Summary

- Introduced an algorithm (SOS) that reasons about social orientations
- Proposed an ad-hoc heuristic for learning social orienations
- Showed empirically that SOS outperforms both Max^n and Paranoid

Discussion and Future Work

- Comparison against human agents or human-devised agents
- Expand the algorithm to be applicable in more domains:
  - Games with elements of chance
  - Games of incomplete information