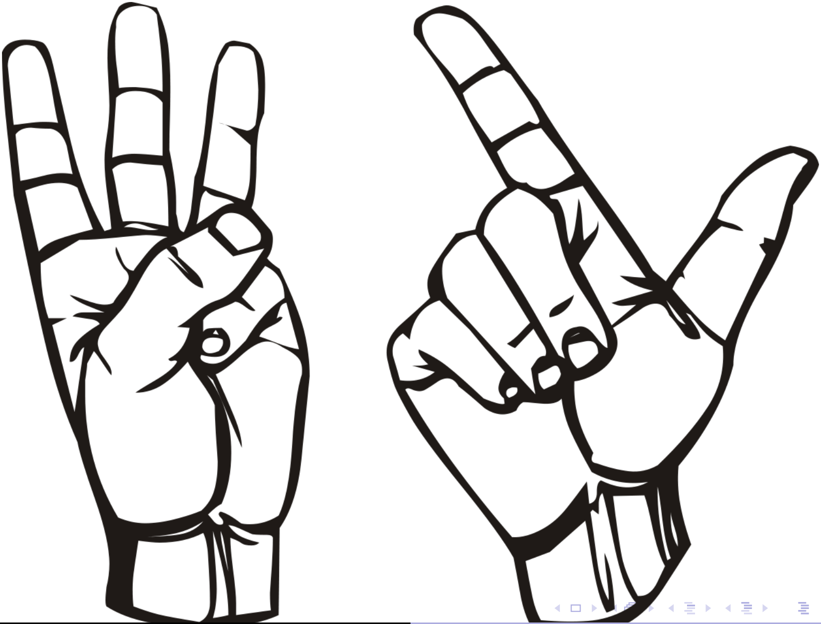
Multiplayer Chopsticks Game

Solving Chopsticks using Adversarial Search Algorithms

Rajat Sharma (B18CSE043)

October 2020

Introduction



Goals

- Formulate Chopsticks as an adversarial search problem
- Develop an evaluation function to score both goal states and intermediate states
- Apply Max^N algorithm to find optimal strategy for each player
- Optimize vanilla Max^N algorithm to allow for greater depth searches

Rules of the Game

The chopsticks game has the following rules:

- Each player begins with one finger raised on each hand. After the first player, turns proceed clockwise.
- On a player's turn, they must either attack or split, but not both.
- To attack, a player uses one of their live hands to strike an opponent's live hand. The number of fingers on the opponent's struck hand will increase by the number of fingers on the hand used to strike.
- To split, a player strikes their own two hands together, and transfers raised fingers from one hand to the other as desired. A move is not allowed to simply reverse one's own hands. You can not spilt 3 with a dead hand. If any hand of any player reaches exactly five fingers, then the hand is killed, and this is indicated by raising zero fingers (i.e. a closed fist).

- A player may revive their own dead hand using a split, as long as they abide by the rules for splitting. However, players may not revive opponents' hands using an attack. Therefore, a player with two dead hands can no longer play and is eliminated from the game.
- If any hand of any player reaches more than five fingers, then five fingers are subtracted from that hand. For instance, if a 4-finger hand strikes a 2-finger hand, for a total of 6 fingers, then 5 fingers are automatically subtracted, leaving 1 finger. Under alternate rules, when a hand reaches 5 fingers and above it is considered a "dead hand".
- A player wins once all opponents are eliminated (by each having two dead hands at once).

Problem Definition

The state of the problem has two main components:

1. State of the hands for each of the N players
2. The current player whose turn it is

Thus it is represented as an $N+1$ tuple, where the first the N elements denote the state of the hands for each player, while the $(N+1)th$ element is the index of the current player. For each player, the state of hands is denoted by a pair of integers $< h_1, h_2 >$, where h_1 and h_2 are the number of fingers on each hand, and $0 \leq h_1, h_2 \leq 4$.

Problem Definition

Thus the state at any point is defined as

$$S = \{(h_{1,1}, h_{2,1}), (h_{1,2}, h_{2,2}), ..., (h_{1,n}, h_{2,n}), p_{curr}\} \tag{1}$$

The initial state is:

$$S_1 = \{(1,1)_1, (1,1)_2, ..., (1,1)_n\} \tag{2}$$

Problem Definition

The transition function (attack) for the problem is defined as:

$$T(S, Attack) = \{(h_{1,1}, h_{2,1}), ..., ((h_{1,k} + h_{j,i}) \mod 5, h_{2,k}), ...$$

$$... \ ,(\mathrm{h}_{1,n}, h_{2,n})\}(3)$$

Where $1 \leq i, k \leq N$ and $1 \leq j, k \leq 2$, $h_{l,k} = (h_{l,k} + h_{j,i}) \mod 5$, and $l = 1$ without loss of generality.

Problem Definition

Suppose player p_i splits $h_{1,i}$, transferring k fingers to hand $h_{2,i}$, $1 \leq k < h_{1,i}$. Then after the split, the status of the hands is $h'_{1,i} = h_{1,i} - k$, $h'_{2,i} = (h_{2,i} + k) \mod 5$ with $h'_{1,i} \neq h_{2,i}, h'_{2,i} \neq h_{1,i}$ and $h'_{1,i}, h'_{2,i} \neq 0$.
The transition function (split) for the problem is defined as:

$$T(S, Split) = \{(h_{1,1}, h_{2,1}), ..., ((h_{i,1} - k), (h_{2,i} + k) \mod 5),$$

$$...,(\mathrm{h}_{1,n}, h_{2,n})\}(4)$$

Problem Definition

The evaluation function is defined as:

$$E(S) = \{(h_{1,1} + h_{2,1}), ..., (h_{1,i} + h_{2,i}), ..., (h_{1,n} + h_{2,n})\} \quad (5)$$

If $(h_{1,i} + h_{2,i}) \neq 0$ and $(h_{1,j} + h_{2,j}) = 0 \forall j \neq i$, then $e_i = 10$ and $e_j = 0 \forall j \neq i$. Thus the utility for a goal state is given by:

$$E(S) = \{e_1 = 0, ..., e_i = 10, ..., e_n = 0\} \quad (6)$$

Non cooperative games

Non cooperative games have the following properties:

1. All players make their moves independently of the other players
2. All alliances, if any are self enforced.

Chopsticks satisfies both of these requirements and hence is a non-cooperative game.

Perfect Information Games

Perfect information games have the following two conditions:

1. All players know the game structure.
2. Each player, when making any decision, is perfectly informed of all the events that have previously occurred.

Chopsticks satisfies both these requirements, which is a trivial observation. For more detailed proof, refer to the report.

Perfect Information Games

Non cooperative, Perfect information games obey the following two theorems, which can help us to define a solution for chopsticks

Theorem

A finite n person non-cooperative game which has perfect information possesses an equilibrium point in pure strategies.

Theorem

Given an n person, non-cooperative, perfect information game in tree form, max^n finds an equilibrium point for the game.

Max^N Algorithm

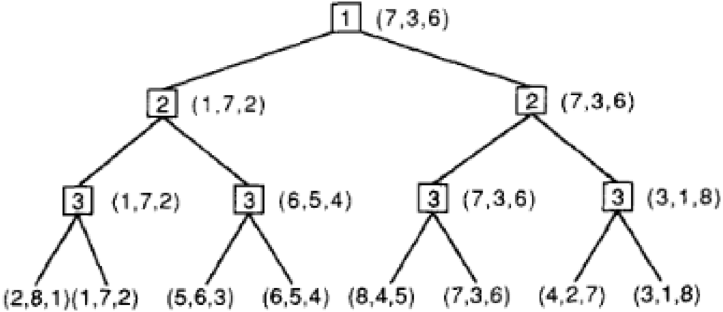


Figure: A sample 3 player Max^N game tree

Max^N Algorithm

Procedure: MaxN(State)

If *State* is Terminal:
    return $Eval(S)$
endif
Initialize $curr \leftarrow State_{n+1}$
Initialize $successor \leftarrow T(State, Attack) \cup T(State, Split)$
Initialize $mx \leftarrow -\infty$
Initialize $util \leftarrow NULL$
For v in *successor*, do:
    $move_{curr} \leftarrow MaxN(v)$
    If $move_{curr} > mx$
        $mx \leftarrow move_{curr}$
        $util \leftarrow move$
    endif
endfor
return *util*

Optimizations Proposed

We have proposed three basic optimizations:

1. In a single search, visit each state only once. If encountered again, either use its memoized value, or use the evaluation function to calculated expected utility.
2. Once a strategy for a given state is calculated, memoize its result and reuse it whenever it is encountered.
3. Search the tree as and when required. Don't precompute the strategies.

Time Complexity Analysis

The following are the time complexities with different implementations:

1. Naive algorithm: $O(B^M)$, where B is the branching factor for the problem, and M is the search depth limit.
2. Visit Once + Memoization: $O(N^2 * 5^{4N})$, where N is the number of players.
3. On Demand Computation: $O(K * N * 5^{2N})$, where K is the number of moves played.

Some Observations and Comparisons

No. of Players (N)	Branching Factor (B)
2	5
3	9
4	14
5	18

Table: Estimated branching factors for different no. of players

No. of Players (N)	$D = 2$	$D = 3$	$D = 5$	$D = 8$
2	0.000s	0.000s	0.011s	0.659s
3	0.000s	0.002s	0.230s	over 1min
4	0.000s	0.008s	1.265s	over 1min
5	0.001s	0.012s	4.726s	over 1min

Table: Time taken by vanilla Max^N for a single move

Some Observations and Comparisons

No. of Players (N)	$D = 2$	$D = 3$	$D = 5$	$D = 8$
2	0.001s	0.001s	0.007s	0.010s
3	0.001s	0.002s	0.048s	0.218s
4	0.001s	0.003s	0.217s	3.596s
5	0.001s	0.003s	0.579s	8.853s

Table: Time taken by Max^N after optimizations for a single move

Worked Out Example

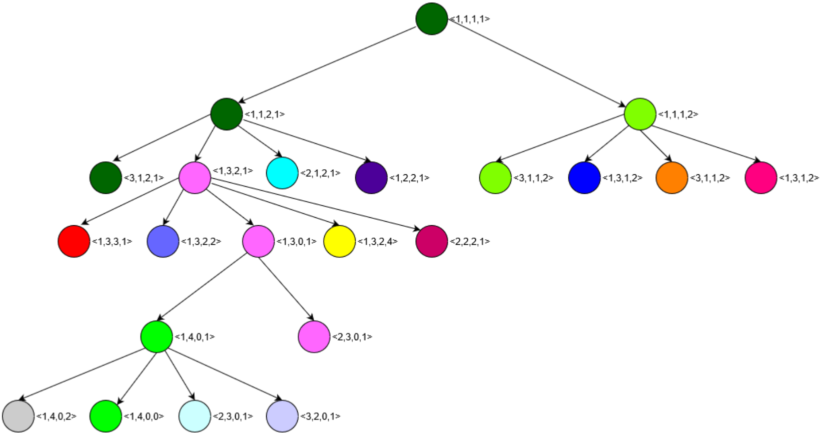


Figure: A worked out example showcasing the use of Max^N for $N = 2$

Worked Out Example

The given example showcases the Max^N algorithm being applied in practice for $N = 2$. For space constraints, the value of N has been kept low, and the entire search tree hasn't been shown. The evaluation function is used to calculate expected utilities at the leaf nodes, which are then propagated upwards by the Max^N algorithm. The colors represent different utilities, and same colors in a parent node indicate that the expected utility for the parent is same as that of the child. In case of a tie, the leftmost node is chosen. At each level, the player whose turn it is tries to maximise his own utility, which guides the working of the algorithm.

Future Prospects

1. Create metrics that can evaluate the optimality of the evaluation function.
2. Explore alternatives to Max^N such as the paranoid algorithm.

1. My parents, my grandparents and my sister for their constant support.
2. My friend and colleague Nivedit Jain (B18CSE039) for his suggestions and review.
3. John Nash, whose life history motivated me to take a topic in game theory.

Finish


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
TASHAKKUR ATU
SUKSAMA
MEHRBANI
GRAZIE
BOLZİN
MERCI
BİYAN
SHUKRIA
JUSPAXAR