Computing Science (CMPUT) 657

Algorithms for Combinatorial Games

Martin Müller

Department of Computing Science
University of Alberta
mmueller@ualberta.ca

Winter 2022

How to Play Sums of Games?

- Special case - impartial games
  - Solved by Nim analysis, MEX rule
  - Or, by more efficient algorithms (Nimbers paper)
- General case - minimax solution
  - Exact solution
  - Approximate solution

Preview - Integer Avoidance Theorem

- How to simplify analysis?
- Which moves are the least urgent to play?
- Integer Avoidance Theorem: Integers are the least urgent to play
- (we will discuss why when we talk about *incentives* of moves)
- So if we have a sum game, with some integers in it, focus on all the other games first
- Practical problem: recognize *when* a game is an integer
- More general idea: *incentive* of a move precisely measures the improvement from making the move

Sums of Hot Games

- Computing a sum directly can be very complex
- Example: $a = 1| - 1, b = 2| - 2, c = 3| - 3, d = 4| - 4$
- Canonical form of sums:
  - $a + b = \{\{3|1\}|\{-1| - 3\}\}$
  - $a + b + c = \{\{\{6|4\}|\{2|0\}\}|\{\{0| - 2\}|\{-4| - 6\}\}\}$
  - $a + b + c + d = \{\{\{10|8\}|\{6|4\}\}|\{\{4|2\}|\{0| - 2\}\}\}|\{\{\{2|0\}|\{-2| - 4\}\}|\{\{-4| - 6\}|\{-8| - 10\}\}\}$
- More CGSuite examples in class notes

Sums of Hot Games (2)

- Worst case example:
- No simplification
- Basically, sum contains the complete search tree
- In many cases we get some simplification
- Example: several small subgames add to a constant
- Example: $\{2|0\} + \{5||3|0\} + \{6|4\} = \{11||9|6\}$
- Why? $\{2|0\} + \{6|4\} = \{2|0\} + 6 + \{0|-2\} = 6$

Sums of Hot Games (3)

- Computing sums often leads to combinatorial explosion
- Need a better approach
  - Compute a best move, or at least a good move in a sum game...
  - ...without adding the games
- One reason for complexity: canonical form of sum is overly general...
- ...if we just want to know the result of **one specific sum** under **alternating play**
- We do not need to know how this sum game behaves in summation with any other game
- How to be more efficient?
- First step: let's revisit minimax search

Minimax Solution

- Can use standard alphabeta search
- Generate all moves in each subgame
- New approach: stop play in integers (discuss)
- Add the scores when all games are played out to integers
- If the score is 0, previous player wins
- Example:
  $\{5||3|2\} + \{10|4|| - 2\} + \{7|6||1|0\}$,
  Left to play

Minimax Solution (2)

- Disadvantage: such search does not use local structure
- Can we solve sum games more efficiently?
- Yes! (Most of the time)
- Algorithm ideas:
  - Prune moves based on incentive
  - Sort moves by *temperature*

Leftscore and Rightscore

- Minimax scores of a game
- Leftscore: Left plays first
- Rightscore: Right plays first
- Example: $5||3|2 + 10|4|| - 2$
  - Leftscore = 9
  - Rightscore = 3
- Can compute (much) faster than by addition

Example - Leftscore and Rightscore with Minimax

- Play $G = 5||3|2 + 10|4|| - 2$, Left moves first
- option 1 (bad): Left moves to $5 + 10|4|| - 2$,
  - Right moves to $5 - 2 = 3$ = final result
- option 2 (correct): Left moves to $5||3|2 + 10|4$
  - Right best move is to $5||3|2 + 4$
    - Left moves to $5 + 4 = \mathbf{9}$ = leftscore(G)
  - Right's mistake: to $3|2 + 10|4$
    - Left best move to $3|2 + 10$,
    - Right to $2 + 10 = 12 > 9$
- Exercise: what is rightscore(G) ?

The Incentive of a Move

- **Incentive**: exact measure of move value
- Improvement in position from making a move from $G = \{G^L | G^R\}$
- Incentive for Left: $G^L - G$
- Incentive for Right: $G - G^R$
- Note the asymmetry in the definition
- For incentives, larger is always better, for both players!

Incentive Examples

- Left incentive: $G^L - G$, Right incentive: $G - G^R$
- Examples:
- $G = 6$, $G^L = 5$
  - Left's incentive = -1
  - Right's incentive does not exist. There is no G^R
- $G = 10|5$, $G^L = 10$, $G^R = 5$
  - Left's incentive = $10 - \{10|5\} = 5|0$
  - Right's incentive = $\{10|5\} - 5 = 5|0$
- $G = *$, $G^L = 0$, $G^R = 0$
  - Left's incentive = $0 - * = *$
  - Right's incentive = $* - 0 = *$
- Are both player's incentives always the same?

Examples - Different Incentives for Both Players

- Are both player's incentives always the same? **No**
- Example 1: $G = 2||1|0$
  - Left's incentive = $G^L - G = 2 - 2\,||\,1\,|\,0 = 2\,|\,1\,||\,0$
  - Right's inc. = $G - G^R = 2\,||\,1\,|\,0 - 1\,|\,0 = \{1, \{2|1\}|0\}$
- Example 2: $G = \uparrow = \{0|*\}$
  - Left move to $G^L = 0$
  - Incentive for Left: $0 - \uparrow = \downarrow$
- Right move to $*$
  - Incentive $G - G^R = \uparrow - * = \uparrow *$

Examples - Different Incentives for One Player

- If one player has several different (incomparable) options
- Then that player has several different (incomparable) incentives
- Example 2: $G = \{2, \{3|1\}|0\}$
- Left option 1: $G^{L1} = 2$
  - Incentive 1 for Left: $2 - G = \{2|0, \pm 1\}$
- Left option 2: $G^{L2} = \{3|1\}$
  - Incentive 2 for Left: $\{3|1\} - G = \{3|1||0\}$
- Right move to 0
  - Incentive $G - G^{R} = G$

Pruning using Incentives

- Theorem (easy): if moves m_1 and m_2 have incentives $I_1 \geq I_2$, then can safely prune m_2.
- Note: it does not matter if the moves are in the same subgame or not
- Leads to a powerful optimal solving algorithm (decomposition search)

Pruning using Incentives

- Given sum game $S = G_1 + ... + G_n$
- Compute incentives of all moves in all subgames
- Incentives can be *computed locally* in each subgame
- Prune: remove all moves with dominated incentives
- Can be *very* effective

Incentives can be Computed Locally

- Given sum game $S = G_1 + ... + G_n$
- Play in subgame G_i to some option G_i^L
- e.g. Left Incentive = $S^L - S =$
  $G_1 + ... + \mathbf{G_i^L} + ... + G_n$
  $-(G_1 + ... + \mathbf{G_i} + ... + G_n)$
  $= G_i^L - G_i$
- Incentive in sum game S = incentive in subgame G_i
- Same for Right incentive
- Other subgames all unchanged, cancel in the subtraction

Pruning using Incentives

- In Go:
  - Endgames typically have strong ordering of incentives
  - Often, one move dominates all others
  - Used in Decomposition Search (Müller 1995, 1999)
- Amazons: effective, but less so than in Go. More positions that have several nondominated moves.
- Bad case: lots of games, incomparable incentives - no pruning
- Worst case: impartial games - no dominated incentives
  - However: can still prune other options with *equal* incentive

Summary

- Part 1: Looked at sums of hot games
- Leftscore and rightscore defined if we know how to stop at an integer
- Difference leftscore - rightscore gives some measure of urgency in playing a game
- Part 2: Incentive is exact measure of value of a move
- It is another game
- Incentives can be computed locally in a subgame, and used for pruning dominated moves, both locally **and globally** in a sum