Unit 2
Adversarial Search
AKA – Game Playing
Introduction

• Why is game playing so interesting from an AI point of view?
  – Game Playing is harder than common searching
    • The search space is (often) too large for complete searches
  – We are facing an unpredictable opponent
    • Games are adversarial search problems
    • Solution is strategy, contingency plan.
    • There are time limits
  – Game playing is considered to be an intelligent activity.
Introduction

- Two-Person games
  - How do we think when we play e.g. Chess?
    - If I move my queen there, then my opponent has to move his knight there and then can I move my pawn there and check mate.
  - We are making some assumptions
    - We want our best move
    - The opponent wants his best move
    - The opponent has the same information as we
    - Our opponent wants to win
A simple introductory game

Consider a game where

- A and B both pick their choice simultaneously
- A either receives money from B (positive numbers) or gives money to B (negative numbers)

<table>
<thead>
<tr>
<th>B chooses B1</th>
<th>B chooses B2</th>
<th>B chooses B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A chooses A1</td>
<td>+3</td>
<td>−2</td>
</tr>
<tr>
<td>A chooses A2</td>
<td>−1</td>
<td>0</td>
</tr>
<tr>
<td>A chooses A3</td>
<td>−4</td>
<td>−3</td>
</tr>
</tbody>
</table>

If you are A, which choice should you make?
If you are B, which choice should you make?
MiniMax

- Games with two players MIN and MAX are a search problem with:
  - Initial state
  - Successor function
  - Terminal test / state
  - Utility function (objective / payoff)
MiniMax

• Utility function
  – Is assumed to be in relation with Max
    • What is good for max is always bad for min.
    • E.g. if max wins then min lose
  – In chess the utility function might be.
    • -1 if min wins
    • 0 for a draw
    • 1 if max wins
  – In other games, e.g. Othello the utility function might be more advanced.
    • utility = number of my pieces – number of his pieces
MiniMax

• Simple Games
  – If we play e.g. TicTacToe or NIM, we can generate a complete search tree
  – Some leafs in the tree will end up with max winning and some with min winning
  – For more complex games a complete search tree is impossible. But that’s a question for the near future.
MiniMax

MAX (x)

MIN (o)

MAX (x)

MIN (o)

TERMINAL

Utility

-1

0

+1
function MiniMax(State s, Event e, boolean isMax)
    State s1 = updateState(s, e)
    if (isLeaf(s1))
        return eval(s1)
    if (isMax)
        highest = -∞
        foreach (Event el in maxmoves(s1))
            tmp = MiniMax(s1, el, !isMax)
            if (tmp > highest)
                highest = tmp
                move = el
        return highest, move
    else
        lowest = ∞
        foreach (Event el in minmoves(s1))
            tmp = MiniMax(s1, el, !isMax)
            if (tmp < lowest)
                lowest = tmp
                move = el
        return lowest, move
MiniMax

• Searching with minimax
  – We use depth first search to reach down to the leafs first
  – When we reach a leaf, we evaluate the current state by a utility function and then returns the result to the parent node.
  – If we are not in a leaf and have evaluated all children.
    • If it, from this state, is max move, then return the highest utility
    • If it is a min move return the lowest utility.
MiniMax

• Example of MiniMax – 1st version of NIM
  – The rules of NIM are as follows
    • We start with a number of sticks in a group
    • On each move, a player must divide a group into two smaller groups
    • Groups of one or two sticks can’t be divided
    • The last player to make a legal move wins
MiniMax

- **NIM Search tree**

```
[6] 6
  1-5  2-4  3-3
  |    |    |
1-1-4 1-2-3 2-2-2
  |    |    |
1-1-1-3 1-1-2-2
  |    |
1-1-1-1-2
```

Max moves
Min moves
Max moves
Min moves
MiniMax

• NIM Search tree

Max moves
Min moves
Max moves
Min moves
MiniMax

- NIM Search tree

```
1-5
  /  \
1-1-4 2-4 3-3
  /   /   /
1-1-1-3 1-2-3 2-2-2
```

Max moves: 1-5, 1-1-4, 1-1-1-3
Min moves: 2-4, 1-2-3, 1-1-2-2
MiniMax

- NIM Search tree

```
      6
     / \
   1-5 2-4 3-3
  /  |  |
1-1-4 1-2-3 2-2-2
  |  |  |
1-1-1-3 1-1-2-2
  |  |
1-1-1-1-2
```

Max moves
Min moves
Max moves
Min moves
MiniMax

- NIM Search tree

Max moves

Min moves
MiniMax

- NIM Search tree

Max moves

Min moves

Max moves

Min moves
MiniMax

• NIM Search tree

6

1-5

2-4

3-3

1-1-4

1-2-3

2-2-2

1-1-1-3

1-1-2-2

1-1-1-1-2

Max moves

Min moves

Max moves

Min moves
MiniMax

- NIM Search tree

Max moves

Min moves

Max moves

Min moves