Simulating Extensive-Form Dilemmas

Justus Hibshman - 11/08/18
Dilemmas:

P1

“Ok.”

“Ugh.”

“Wonderful!”

“Ok.”

P2

“Ugh.”

“Ugh.”

“Wonderful!”
Dilemmas:

“Ok.”
“Ugh.”

“Wonderful!”
“Ok.”

“Ugh.”
“Wonderful!”
Do Dilemmas Occur Naturally?
Kernel: MiniMax with DFS

```
Def MiniMax(root)
    If(root.children.size() == 0)
        Return root.preferences
    best_outcome = -inf
    result = Null
    For child in root.children
        child_result = MiniMax(child)
        If(child_result[root.player_id] > best_outcome)
            best_outcome = child_result[root.player_id]
            result = child_result
    Return result
```
Complexity

Number of players: \( p \)
Number of vertices: \( v \)
Number of edges: \( e = v - 1 \) because it's a tree.

MiniMax with DFS: \( O(p(v + e)) = O(pv) \)

Game Tree Generation: \( O(pv) \)

Checking Optimality of Result: \( O(pv) \)
Implementation

● **DFS: Boost Graph Library (C++)**
  ○ Overload “DFS Visitor” class

● **Game Generation**
  ○ Custom C++ code
  ○ Boost graph format
  ○ Players’ preferences are strict orderings (all >, no >=).
  ○ “Player i prefers A over B” tells you nothing about Player j’s preferences.
Experiment Setup

Run 1000 trials for each parametrization:

- Vary number of players from 2 to 5
- Vary number of game tree nodes from 10 to 1,000,000
- Balanced Trees
  - Vary degree from 2 to 32
- “Chain” Trees
  - Every decision node has one stop-edge leading to a final outcome and one continue-edge.
  - Example:
2 Player Balanced Tree Games

Degree:
- 2
- 4
- 8
- 16
- 32

Percent Outcomes Pareto Optimal (1000 trials)

Size of Game (# Tree nodes)
3 Player Balanced Tree Games

Degree:
- 2
- 4
- 8
- 16
- 32
4 Player Balanced Tree Games

Degree:
- 2
- 4
- 8
- 16
- 32

Percent Outcomes Pareto Optimal (1000 trials)

Size of Game (# Tree nodes)
5 Player Balanced Tree Games

Percent Outcomes Pareto Optimal (1000 trials)

Size of Game (# Tree nodes)

Degree:
- 2
- 4
- 8
- 16
- 32
2 Player Chain Games

Percent Outcomes Pareto Optimal (1000 trials)

Size of Games (# Tree Nodes)
Runtime Scaling with Game Size

Average Runtime of Experiment (Unknown Units)

Number of Nodes in Game Tree

- 2 Players Runtime
- 3 Players Runtime
- 4 Players Runtime
- 5 Players Runtime
Runtime Scaling with Number of Players

![Graph showing the relationship between Total Experiment Runtime Percent Increase and Number of Players. The graph is a straight line starting at (2, 1) and ending at (5, 1.4).]
Future Work

● Parallelize DFS
  ○ May require locks in DFS visitor?

● Parallelize Game Generation
  ○ Need to learn MPI?

● Rather than running a single test in parallel, run different tests on different processes? (Less interesting. Might waste memory.)