Simulating Dilemmas

Justus Hibshman - 12/10/18
Players/Agents $i=1...n$

Consequence set $C = \{c_1, ..., c_m\}$

Preferences $\succeq_i$

Strategy Set $S = S_1 \times ... \times S_n$

$f: S \rightarrow C$
Extensive Form Games

Strategies are composed of decisions at tree nodes.

Example:

\[ S = \{A\} \]

\[ S = \{C, F\} \]
Dilemmas

Assumptions:

- All players are rational.
- All players know the other players’ preferences.
- Everything a player cares about has been encapsulated in the preference relation.
- These facts are “Common Knowledge.”
  - Basically means “I know that you know that I know…”
Dilemmas

P1

≿

≿

≿

P2

≿

≿

≿
Dilemmas

P1

P2

A → B → C → D
Dilemmas
Dilemmas

P1

P2

≿ ≿ ≿ ≿

≿ ≿ ≿ ≿
Dilemmas

Inevitable Result
Dilemmas

Inevitable Result
Do Dilemmas Occur Naturally?

- Big Trees
- Random Preferences
```python
def mini_max(node):
    if node.is_leaf():
        return node.consequence
    result = null
    for child in node.children:
        child_consequence = mini_max(child)
        if result is null or node.player.prefers(child_consequence, result):
            result = child_consequence
    return result
```
Complexity

Number of players: \( p \)

Number of vertices: \( v \)

Number of edges: \( e = v - 1 \) because its a tree.

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

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

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

- DFS: Boost Graph Library (C++)
  - Overload “DFS Visitor” class
- Game Generation
  - Custom C++ code
  - Boost graph format
Enhanced Implementation

- Custom Graph Format (C++)
  - Adjacency List Implementation
- Custom Tree Traversal Code
- Parallelization done with pthreads
- ~550 Lines of Code
Enhanced Implementation

Basic idea:

- Available worker-thread pool
- Workers do sub-traversals
- Any worker can assign sub-traversal to other workers
- When a worker finishes:
  - Wake up the thread waiting on results
  - Re-enter the pool
Experiment Setup

Run 100 trials for each parametrization:

- Number of players: 4
- Vary number of game tree nodes from 10 to 10,000,000
- Balanced Trees (degree of 8)
- “Chain” Trees
  - Every decision node has one stop-edge leading to a final outcome and one continue-edge.
  - Example:
    [Diagram of a chain tree]
Results

Frequency of Dilemmas in "Chain" Trees

Fraction Dilemma

Num Nodes

10 100 1000 10000 100000 1000000
Results

Frequency of Dilemmas in Balanced Trees (deg = 8)
Results

"Chain" Trees -- Traversal Scaling

Runtime For 100 Trials (in ms)

Num Nodes
"Chain" Trees -- Traversal Scaling

[Diagram showing runtime for 100 trials in ms vs. number of nodes for different thread counts (16, 8, 4, 2, 1) and Boost-Based]
Results

Balanced Trees -- Traversal Scaling

- 16 Threads
- 8 Threads
- 4 Threads
- 2 Threads
- 1 Thread

Runtime For 100 Trials (in ms)

Num Nodes
Results

Balanced Trees -- Traversal Scaling

Runtime For 100 Trials (in ms)

Num Nodes

- 16 Threads
- 8 Threads
- 4 Threads
- 2 Threads
- 1 Thread
- Boost-Based
Reference

- Boost C++ Libraries
  - https://www.boost.org/