Monte Carlo Tree Search and the Expected Outcome Heuristic

Andrew Gelfand
8/24/2011
Tree Search

- **Ex: Tile game**
  - State: location of tiles
  - Actions: move blank left, right, up, down

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```
Informed Search

- Evaluate each node using function $f(x)$
- **Strategy**: Expand node with smallest $f(x)$
- **Ex**: A* Search
  \[ f(x) = g(x) + h(x) \]

  - $g(x)$: Cost to reach node $x$
  - $h(x)$: Estimate of cost to goal from node $x$

**Sample Heuristics**
- $h(x) = \#$ misplaced tiles
- $h(x) = $ Manhattan distance
Expected Outcome (EO) Heuristic

- Identifying 'good' evaluation function is hard

Expected Outcome (EO) [Abramson 90]

- "Elegant, crisply defined, easily estimable, and above all, domain independent"

\[
EO(x) = \sum_{k=1}^{K} g(x_k) \cdot p(x_k)
\]

- \( K \) # of leaves in subtree rooted at \( x \)
- \( g(x_k) \) cost of leaf node \( x_k \)
- \( p(x_k) \) probability of reaching node \( x_k \) given random actions i.e. under a 'Plinko' expansion strategy

Note: \( p(x_k) \neq 1/K \) (in general)
Monte Carlo Evaluation

- Estimate $p(x_i)$ by running simulations from $x$
  - Simulations must run quickly!
- Recursively build search tree
- Assuming simulations are iid

$$EO_n(x) = \frac{1}{n} \sum_{i=1}^{n} g(x^{(i)}) = EO_{\infty}(x) + R$$

- Evaluation of $x$ after $n$ simulations
- Cost of $i^{th}$ simulation
- Evaluation of $x$ after $\infty$ simulations
- $R \sim N\left(0, \sigma / \sqrt{n}\right)$ (Error Term)
Evaluating EO

- Is EO a good evaluation function?
  - Works well in some games (e.g. Othello)
  - “random” actions can be a bad assumption
    - Assume perfect play by opponent => cautious moves
    - Assume random play by opponent => ??

- How big must \( n \) be?
  - \( \sigma \) is unknown, so tough to say in general
  - \( n > 1000 \) in game of Go [Bouzy & Helmstetter 06]
    - Time consuming to get to a reasonable depth!
Extending MCE

- Combine search with MCE
  - Greater depth given fewer simulations
  - [Bouzy 06] Iteratively deepen and prune
  - [Persson 06] αβ search w/ lazy evaluations

- Learn the EO
  - EO is expectation, so fit $f(x)$ by minimizing SSE
    - Ex: Othello
      - $m$ random initial board configurations
      - Sample to get $\hat{EO}$ from each configuration
      - Fit function of form $f(x) = \sum_j w_j \phi_j(x)$
Monte Carlo Tree Search

- Don't prune; Gradually build search tree

1: Select
2: Expand
3: Simulate
4: Update/Backprop

For each node maintain:
- $v_i$ - value of node $x_i$
- $n_i$ - # times node $x_i$ visited
MCTS – 1) Selection Step

- Apply *selection strategy* until unexplored node
- Balance exploration and exploitation

Strategies in literature:

- OMC (Objective Monte Carlo) [Chaslot et al. 06]
  - Selects action proportional to the prob. of the action being better than the current best action
- PBBM (Probability to be Better than Best Move) [Coulom 06]
  - Similar to OMC, but accounts for uncertainty
MCTS – 1) Selection Step

- UCT (Upper Confidence Bounds) [Kocsis & Szepesvari 06]
  - Based upon Upper Confidence Bounds (UCB) method for Multi-armed bandit problems
  - Select child $k$ of node $p$ according to:

$$k \in \arg \max_{i \in I} v_i + C \cdot \sqrt{\frac{\ln(n_p)}{n_i}}$$

- $I$ set of nodes reachable from node $x_p$
- $v_i$ current value of $x_i$
- $n_i$ visit count of node $x_i$
- $n_p$ visit count of node $x_p$
- $C$ experimentally tuned coefficient
MCTS – 2) Expansion Step

- **Typical Strategy:**
  - Expand one node per simulated game
  - Add expanded node to the MCTS tree
  - Expanded node is typically 1\textsuperscript{st} node encountered not in memory
MCTS – 3) Simulation Step

- Selects actions randomly from the expanded node until leaf node is reached
- **Choice:** Expand Randomly or use Heuristic?
  - **Tradeoff:**
    - If purely random, actions can be 'poor' causing unrealistic simulations (too much exploration)
    - If too deterministic, selected action for a given node may be same (too much exploitation)
    - Either way, must be fast!!!
MCTS – 4) Update/Backprop Step

- Propagates cost of simulation to those nodes traversed in MCTS tree

- Ex: Average of all simulations through node

\[ v_i = \frac{1}{n_i} \sum_{k \in K} g(x^{(k)}) \]

- Set of simulations through node \( x_i \)
- Cost to reach a terminal node in simulation \( k \)