Learning to Prune Dominated Action Sequences in Online Black-box Planning

Yuu Jinnai    Alex Fukunaga

The University of Tokyo
Black-box Planning in Arcade Learning Environment

• What a human sees

Arcade Learning Environment
(Bellemare et al. 2013)
Black-box Planning in Arcade Learning Environment

- What the computer sees

Arcade Learning Environment
(Bellemare et al. 2013)
General-purpose agents have many irrelevant actions

- The set of actions which are “useful” in each environment (= game) is a subset of the available action set in the ALE
- Yet an agent has no prior knowledge regarding which actions are relevant to the given environment in black-box domain

<table>
<thead>
<tr>
<th>Neutral</th>
<th>Neutral + fire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>Up + fire</td>
</tr>
<tr>
<td>Up-left</td>
<td>Up-left + fire</td>
</tr>
<tr>
<td>Left</td>
<td>Left + fire</td>
</tr>
<tr>
<td>Down-left</td>
<td>Down-left + fire</td>
</tr>
<tr>
<td>Down</td>
<td>Down + fire</td>
</tr>
<tr>
<td>Down-right</td>
<td>Down-right + fire</td>
</tr>
<tr>
<td>Right</td>
<td>Right + fire</td>
</tr>
<tr>
<td>Up-right</td>
<td>Up-right + fire</td>
</tr>
</tbody>
</table>

Available action set in the ALE (18 actions)  Actions which are useful in the environment
Two ways of domain description

- Transparent model domain (e.g. PDDL)
- Black-box domain
Transparent Model Domain

Input: initial state, goal condition, action set is described in logic (e.g. PDDL)
- Easy to compute relevant action
- Possible to deduce which actions are useful

Init: ontable(a), ontable(b), clear(a), clear(b)
Goal: on(a, b)
Action:
  Move(b, x, y)
  Precond: on(b, x), clear(x), clear(y)
  Effect: on(b, y), clear(x), ¬on(b, x), ¬clear(y)

Example: blocks world

Initial state

Goal condition

Input: initial state, goal condition, action set is described in logic (e.g. PDDL)
- Easy to compute relevant action
- Possible to deduce which actions are useful

Init: ontable(a), ontable(b), clear(a), clear(b)
Goal: on(a, b)
Action:
  Move(b, x, y)
  Precond: on(b, x), clear(x), clear(y)
  Effect: on(b, y), clear(x), ¬on(b, x), ¬clear(y)

Example: blocks world

Initial state

Goal condition
Black-box Domain

- Domain description in Black-box domain:
  - $s_0$: initial state (bit vector)
  - $suc(s, a)$: (black-box) successor generator function returns a state which results when action $a$ is applied to state $s$
  - $r(s, a)$: (black-box) reward function (or goal condition)

→ No description of which actions are valid/relevant
Arcade Learning Environment (ALE): A Black-box Domain (Bellemare et al. 2013)

- Domain description in the ALE:
  - State: RAM state (bit vector of 1024 bits)
  - Successor generator: Complete emulator
  - Reward function: Complete emulator
• Domain description in the ALE:
  • 18 available actions for an agent
  • **No description of which actions are relevant/required**
  • Node generation is the main bottleneck of walltime (requires running simulator)
Two Lines of Research in the ALE  
(Bellemare et al. 2013)

- **Online planning setting** (e.g. Lipovetzky et al. 2015)
  
  An agent runs a simulated lookahead each $k (= 5)$ frames and chooses an action to execute next (**no prior learning**)

- **Learning setting** (e.g. Mnih et al. 2015)
  
  An agent generates a reactive controller for mapping states into actions

---

**We focus on Online planning setting for this talk**
(applying our method to RL is future work)
Online Planning on the ALE
(Bellemare et al 2013)

For each planning iteration (= planning episode)
1. Run a simulated lookahead with a limited amount of computational resource (e.g. # of simulation frames)
2. Choose an action which leads to the best accumulated reward
Online Planning on the ALE  
(Bellemare et al 2013)

For each planning iteration (= planning episode)
1. Run a simulated lookahead with a limited amount of computational resource (e.g. # of simulation frames)
2. Choose an action which leads to the best accumulated reward
Online Planning on the ALE
(Bellemare et al 2013)

For each planning iteration (= planning episode)
1. Run a simulated lookahead with a limited amount of computational resource (e.g. # of simulation frames)
2. Choose an action which leads to the best accumulated reward
For each planning iteration (= planning episode)

1. Run a simulated lookahead with a limited amount of computational resource (e.g. # of simulation frames)

2. Choose an action which leads to the best accumulated reward
General-purpose agents have many irrelevant actions

- The set of actions which are “useful” in each environment (= game) is a subset of the available action set in the ALE

<table>
<thead>
<tr>
<th>Available action set in the ALE (18 actions)</th>
<th>Actions which are useful in the environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutral</td>
<td>Neutral + fire</td>
</tr>
<tr>
<td>Up</td>
<td>Up + fire</td>
</tr>
<tr>
<td>Up-left</td>
<td>Up-left + fire</td>
</tr>
<tr>
<td>Left</td>
<td>Left + fire</td>
</tr>
<tr>
<td>Down-left</td>
<td>Down-left + fire</td>
</tr>
<tr>
<td>Down</td>
<td>Down + fire</td>
</tr>
<tr>
<td>Down-right</td>
<td>Down-right + fire</td>
</tr>
<tr>
<td>Right</td>
<td>Right + fire</td>
</tr>
<tr>
<td>Up-right</td>
<td>Up-right + fire</td>
</tr>
</tbody>
</table>
General-purpose agents have many irrelevant actions

- The set of actions which are “useful” in each environment (= game) is a subset of the available action set in the ALE.
- The set of actions which are “useful” in each state in the environment is a smaller subset.

<table>
<thead>
<tr>
<th>Neutral</th>
<th>Neutral + fire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>Up + fire</td>
</tr>
<tr>
<td>Up-left</td>
<td>Up-left + fire</td>
</tr>
<tr>
<td>Left</td>
<td>Left + fire</td>
</tr>
<tr>
<td>Down-left</td>
<td>Down-left + fire</td>
</tr>
<tr>
<td>Down</td>
<td>Down + fire</td>
</tr>
<tr>
<td>Down-right</td>
<td>Down-right + fire</td>
</tr>
<tr>
<td>Right</td>
<td>Right + fire</td>
</tr>
<tr>
<td>Up-right</td>
<td>Up-right + fire</td>
</tr>
</tbody>
</table>

Available action set in the ALE (18 actions)  Actions which are useful in the environment  Actions which are useful in the state
• Generated duplicate nodes can be pruned by duplicate detection
• However, in simulation-based black-box domain node generation is the main bottleneck of the walltime performance
→ By pruning irrelevant actions we should make use of the computational resource more efficiently
Dominated action sequence pruning (DASP)

- **Goal:** Find action sequences which are useful in the environment (for simplicity we explain using action sequence of length=1)
- Prune redundant actions in the course of online planning
- Find a minimal action set which can reproduce previous search graphs and use the action set for the next planning episode
Dominated action sequence pruning (DASP)

Action set available to the agent
{Up, Down, Up+Fire, Down+Fire}

Minimal action set
{Up, Down}
DASP: Find a minimal action set

- Algorithm: Find a minimal action set $\Delta$

search graphs in previous episodes
**DASP: Find a minimal action set**

- Algorithm: Find a minimal action set $A$

1. $v_i \in V$ corresponds to action $i$ in hypergraph $G = (V, E)$.

![Diagram showing a hypergraph and search graphs](image_url)
DASP: Find a minimal action set

· Algorithm: Find a minimal action set $A$

1. $v_i \in V$ corresponds to action $i$ in hypergraph $G = (V, E)$.

   $e(v_0, v_1, ..., v_n) \in E$ iff there is one or more duplicate search nodes generated by all of $v_0, v_1, ..., v_n$ but not by any other actions.
DASP: Find a minimal action set

- Algorithm: Find a minimal action set $A$

1. $v_i \in V$ corresponds to action $i$ in hypergraph $G = (V, E)$.
   
   $e(v_0, v_1, ..., v_n) \in E$ iff there is one or more duplicate search nodes generated by all of $v_0, v_1, ..., v_n$ but not by any other actions.

2. Add the minimal vertex cover of $G$ to $A$

$A = \{\text{Up, Down}\}$

Hypergraph $G$

search graphs in previous episodes
Algorithm: Find a minimal action set $A$

1. $v_i \in V$ corresponds to action $i$ in hypergraph $G = (V, E)$.
   $e(v_0, v_1, \ldots, v_n) \in E$ iff there is one or more duplicate search nodes generated by all of $v_0, v_1, \ldots, v_n$ but not by any other actions.

2. Add the minimal vertex cover of $G$ to $A$

- $A = \{\text{Up}, \text{Down}\}$

![Graph with actions Up and Down and corresponding hypergraph $G$]
Experimental Result: acquired minimal action set

- DASP finds and uses a minimal action set at each planning episode except for the first 12 planning episodes.
- Restricted action set: hand-coded set of minimal actions for each game.
Problem of DASP

- DASP is a binary classifier: to prune or not to prune
- Most of the actions are only **conditionally effective**

1. **FIRE** action may be useful only if the agent has a sword or a bomb. Such actions may be preemptively pruned before encountering a context it becomes useful. DASP only guarantees that the action set reproduce search graphs of *previous* planning episodes.

2. **LEFT** action may be meaningless if there is a wall on the left of the agent

   DASP may not prune conditionally ineffective actions

→ Should prune actions in the context of the current planning episode!
Dominated action sequence avoidance (DASA)

- **Goal:** Find actions which are useful in the planning episode
- Let $p(a, t)$ be the ratio of new nodes action $a$ generated at $t$-th planning episode.
- From $p(a, t)$ we estimate $p^*(a, t)$: probability of action $a$ generating a new node at $t+1$-th planning episode.

\[
p^*(a, 0) = 1
\]
\[
p^*(a, t+1) = \frac{p(a, t) + \alpha p^*(a, t)}{1 + \alpha}
\]

- At $t$-th planning episode, for each node expansion, agent applies action $a$ with probability $P(a, t)$

\[
P(a, t) = (1 - \epsilon) s(p^*(a, t)) + \epsilon
\]

where $s$ is a smoothing function (e.g. sigmoid), $\epsilon$ is a minimal probability to apply action $a$. 
Experimental Evaluation

- Compared scores achieved on 53 games in the ALE
- Applied DASP and DASA to breadth-first search variants
  - p-IW(1) (Shleyfman et al. 2016), IW(1) (Lipovetzky et al. 2012), BrFS (breadth-first search)
- Limited the number of node generation per planning episode to 2000 (excluding “reused” nodes generated in previous planning episode)

- **DASA2**: DASA applied to action sequence of length = 2
- **DASA1**: DASA applied to action sequence of length = 1
- **DASP1**: DASP applied to action sequence of length = 1
- **default**: Use all available actions in the ALE (18 actions)
- **restricted**: A minimal action set required to solve the game (hard-coded by a human for each game)
**Experimental result: Score**

- **DASA2** had the best coverage for all five settings
  - p-IW(1) (400gend) configuration:
    - Limited the number of node generation to 400. **DASA2** outperformed the other methods.
  - p-IW(1) (extend) configuration:
    - Added two spurious buttons with no effect. **DASA2** outperformed the other methods.

<table>
<thead>
<tr>
<th></th>
<th>DASA2</th>
<th>DASA1</th>
<th>DASP1</th>
<th>default</th>
<th>restricted</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-IW(1)</td>
<td>22</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>p-IW(1) (400gend)</td>
<td><strong>24</strong></td>
<td>14</td>
<td>6</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>IW(1)</td>
<td>22</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>BrFS</td>
<td>18</td>
<td>11</td>
<td>11</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>p-IW(1) (extend)</td>
<td><strong>39</strong></td>
<td>22</td>
<td>19</td>
<td>16</td>
<td>-</td>
</tr>
</tbody>
</table>

Coverage = #Games where each method (column) scored the best among the methods (in each row/configuration)
Experimental Results: Depth of the search

- Compared the number of node expansion and the depth of the search tree using p-IW(1)
- The result indicates that **DASA2** is successfully exploring larger and deeper state-space

<table>
<thead>
<tr>
<th></th>
<th>DASA2</th>
<th>DASA1</th>
<th>DASP1</th>
<th>default</th>
<th>restricted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expanded</td>
<td><strong>254.9</strong></td>
<td>191.1</td>
<td>119.9</td>
<td>119.6</td>
<td>234.0</td>
</tr>
<tr>
<td>Depth</td>
<td><strong>82.8</strong></td>
<td>59.5</td>
<td>34.6</td>
<td>34.1</td>
<td>40.8</td>
</tr>
</tbody>
</table>

Expanded = the average number of node expansion
Depth = the depth of the search tree
• Proposed DASP and DASA, methods to avoid redundant actions in Black-box Domain
• We experimentally evaluated DASP and DASA in the ALE
• Showed that by avoiding redundant actions an agent can search deeper and achieved higher score

Lesson:
• Avoiding redundant action sequences avoids generating duplicate states, a bottleneck in simulation-based black-box domains

Future Work
• Apply DASA in RL (currently working on this)
• Extract more information from the domain
Appendix slides
Experimental Result: number of pruned actions

- Pruned many actions (#available action = 18)
- Restricted action set: a minimal action set required (hard-coded by a human for each game)
IW(1) Example: Tick-Tack-Toe

novelty = 1
IW(1) Example: Tick-Tack-Toe

novelty = 1

novelty = 1
IW(1) Example: Tick-Tack-Toe

novelty = 1

novelty = 1
IW(1) Example: Tick-Tack-Toe

novelty = 1

novelty = 1

novelty = 1

novelty = 1
IW(1) Example: Tick-Tack-Toe

novelty = 1

novelty = 1

novelty = 1

novelty = 1
IW(1) Example: Tick-Tack-Toe

- Aggressive pruning strategy