Neural Network Heuristic Functions for Classical Planning: Reinforcement Learning and Comparison to Other Methods

Patrick Ferber\textsuperscript{1,3}  Florian Geißer\textsuperscript{2}  Felipe Trevizan\textsuperscript{2}  
Malte Helmert\textsuperscript{1}  Jörg Hoffmann\textsuperscript{3}

\textsuperscript{1}University of Basel, Switzerland 
\textsuperscript{2}Australian National University, Australia 
\textsuperscript{3}Saarland University, Germany 

Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (2021)
Motivation

Silver et al. (2016)
Silver et al. (2017)
Silver et al. (2018)

Agostinelli et al. (2019)
Neural Networks as Planning Heuristics

*per-instance* heuristics

- Ferber, Helmert, and Hoffmann (2020)
- Yu, Kuroiwa, and Fukunaga (2020)

*per-domain* heuristics

- Shen, Trevizan, and Thiébaux (2020)
- Rivlin, Hazan, and Karpas (2020)
- Karia and Srivastava (2021)
Neural Networks as Planning Heuristics

**per-instance heuristics**

- Ferber, Helmert, and Hoffmann (2020)
- Yu, Kuroiwa, and Fukunaga (2020)

**per-domain heuristics**

- Shen, Trevizan, and Thiébaux (2020)
- Rivlin, Hazan, and Karpas (2020)
- Karia and Srivastava (2021)
Neural Networks as Planning Heuristics

**per-instance heuristics**

- Ferber, Helmert, and Hoffmann (2020)
- Yu, Kuroiwa, and Fukunaga (2020)

**per-domain heuristics**

- Shen, Trevizan, and Thiébaux (2020)
- Rivlin, Hazan, and Karpas (2020)
- Karia and Srivastava (2021)
Contributions

• three *per-instance* RL based heuristics
  • learned from scratch
  • only state as input
  • prove convergence to $h^*$

• comparison between state-of-the-art
  • neural network heuristics
  • model-based heuristics
Finite-Domain Representation (Helmert, 2009)
Finite-Domain Representation (Helmert, 2009)

\[ \Pi = \langle V, O, I, g \rangle \]

- \( V = \{ \text{on}, \text{on}, \text{on} \} \)
- \( \text{dom}(\text{on}) = \{ \text{on}, \text{on}, \text{on} \} \)
- \( O = \{ \text{move from X to Y} \} \)
- \( I = \) 
- \( g = \{ \text{on} \mapsto \text{on} \} \)
Progression & Regression

move \textcolor{red}{\textbullet} from \textcolor{green}{\textbullet} to

\textit{pre} : \{\textcolor{green}{\textbullet} \mapsto \textcolor{red}{\textbullet} \text{ on } \textcolor{green}{\textbullet} \} \\
\textit{eff} : \{\textcolor{green}{\textbullet} \mapsto \textcolor{green}{\textbullet} \text{ on } \textcolor{red}{\textbullet} \}
Progression & Regression

move  from  to

\[ pre : \{ \text{on} \mapsto \text{on} \} \]
\[ eff : \{ \text{on} \mapsto \text{on} \} \]

Progression  Regression
Progression & Regression

move from \( \ell_1 \) to \( \ell_2 \)

\[ \text{pre} : \{ \ell_1 \rightarrow \text{on} \ \ell_2 \} \]

\[ \text{eff} : \{ \ell_1 \leftarrow \text{on} \ \ell_2 \} \]
Residual Network (He et al., 2016)
def train(Π, NN, t_{train}):
    D = Buffer()

    while time() ≤ t_{train}:
        p = regression random walk(Π)
        s = complete to state (p)
        π = GBFS+NN(s)

        for s′ ∈ π:
            D.push(s′, distance(s′, goal(Π), π)

        NN = train(NN, D)

inspired by Bootstrap Learning of Arfaee, Zilles, and Holte (2011)
Bootstrapping

```python
1  def train(Π, NN, t_train):
2      D = Buffer()
3      L = 5
4      while time() ≤ t_train:
5          p = regression random walk(Π, max_length=L)
6          s = complete to state (p)
7          π = GBFS+NN(s)
8
9          for s' ∈ π:
10             D.push(s', distance(s', goal(Π), π)
11          if frequently solves s:  L = 2 * L
12          NN = train(NN, D)
```

inspired by Bootstrap Learning of Arfaee, Zilles, and Holte (2011)
Bootstrapping

```
1  def train(Π, NN, t_train, t_search):
2      D = Buffer()
3      L = 5
4      while time() ≤ t_train:
5          p = regression random walk(Π, max_length=L)
6          s = complete to state (p)
7          π = GBFS+NN(s, timeout=t_search)
8          if not π: continue
9          for s′ ∈ π:
10             D.push(s′, distance(s′, goal(Π), π)
11             if frequently solves s:  L = 2 * L
12             NN = train(NN, D)
```

inspired by Bootstrap Learning of Arfaee, Zilles, and Holte (2011)
Bootstrapping to Predict Expansions

```python
1  def train(Π, NN, t_{train}, t_{search}):
2      D = Buffer()
3      L = 5
4      while time() ≤ t_{train}:
5          p = regression random walk(Π, max_length=L)
6          s = complete to state (p)
7          expansions = GBFS+NN(s, timeout=t_{search})
8          if not π:
9              continue
10             D.push(s, expansions)
11
12             if frequently solves s: L = 2 * L
13             NN = train(NN, D)
```

inspired by Bootstrap Learning of Arfaee, Zilles, and Holte (2011)
Approximate Value Iteration

1 def train(Π, NN, t_{\text{train}}):
2 \quad D = \text{Buffer}()
3 \quad \text{while } \text{time}() \leq t_{\text{train}}:
4 \quad \quad p = \text{regression random walk}(\Pi)
5 \quad \quad s = \text{complete to state}(p)
6 \quad \quad h = \text{BellmanUpdate}(s, NN)
7 \quad \quad D.\text{push}(s, h)
8 \quad \quad NN = \text{train}(NN, D)
9
10 def BellmanUpdate(s, NN):
11 \quad \text{return } 1 + \min_{s' \in \text{succ}(s)} NN(s')
Algorithms

- $h^{\text{Boot}}$: Bootstrapping
- $h^{\text{BExp}}$: Bootstrapping with expansions
- $h^{\text{AVI}}$: Approximate value iteration
- $h^{\text{SL}}$: Ferber, Helmert, and Hoffmann (2020)
- $h^{\text{HGN}}$: Shen, Trevizan, and Thiébaux (2020)
- $h^{\text{FF}}$: Hoffmann and Nebel (2001)
- $LAMA$: Richter and Westphal (2010)
Benchmarks (Ferber, Helmert, and Hoffmann, 2020)

- Blocksworld
- Depots
- Grid
- NPuzzle
- Pipesworld-NT
- Rovers
- Scanalyzer
- Storage
- Transport
- Visitall
Benchmarks (Ferber, Helmert, and Hoffmann, 2020)

- Blocksworld
- Depots
- Grid
- NPuzzle
- Pipesworld-NT
- Rovers
- Scanalyzer
- Storage
- Transport
- Visitall

<table>
<thead>
<tr>
<th></th>
<th>Trivial</th>
<th>Moderate</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocksworld</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Depots</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPuzzle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipesworld-NT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rovers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scanalyzer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visitall</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Validation (Moderate Tasks)

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h^{Boot}$</th>
<th>+V</th>
<th>$h^{BExp}$</th>
<th>+V</th>
<th>$h^{AVI}$</th>
<th>+V</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>0.0</td>
<td>+18.0</td>
<td>0.0</td>
<td>+0.0</td>
<td>0.0</td>
<td>+0.0</td>
</tr>
<tr>
<td>depots</td>
<td>31.7</td>
<td>+28.6</td>
<td>17.1</td>
<td>+15.0</td>
<td>43.7</td>
<td>+11.0</td>
</tr>
<tr>
<td>grid</td>
<td><strong>100.0</strong></td>
<td>+0.0</td>
<td><strong>100.0</strong></td>
<td>+0.0</td>
<td>51.0</td>
<td>+0.0</td>
</tr>
<tr>
<td>npuzzle</td>
<td>27.0</td>
<td>+1.0</td>
<td>0.0</td>
<td>+0.0</td>
<td>1.0</td>
<td>+0.0</td>
</tr>
<tr>
<td>pipes-nt</td>
<td>36.2</td>
<td>+21.6</td>
<td><strong>51.2</strong></td>
<td>+17.2</td>
<td>21.4</td>
<td>+28.8</td>
</tr>
<tr>
<td>rovers</td>
<td>36.5</td>
<td>+11.7</td>
<td>15.2</td>
<td>+6.6</td>
<td>34.2</td>
<td>+10.8</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>33.3</td>
<td>+0.0</td>
<td>59.7</td>
<td>+11.0</td>
<td><strong>66.7</strong></td>
<td>+0.6</td>
</tr>
<tr>
<td>storage</td>
<td><strong>89.0</strong></td>
<td>+0.0</td>
<td>61.0</td>
<td>-3.5</td>
<td>67.0</td>
<td>+2.5</td>
</tr>
<tr>
<td>transport</td>
<td><strong>83.8</strong></td>
<td>+16.2</td>
<td>79.5</td>
<td>+20.5</td>
<td>70.0</td>
<td>+17.5</td>
</tr>
<tr>
<td>visitall</td>
<td><strong>17.0</strong></td>
<td>+38.3</td>
<td>0.0</td>
<td>+0.0</td>
<td>0.0</td>
<td>+0.0</td>
</tr>
</tbody>
</table>

Table: Performance of our algorithms without validation and performance change due to validation (+V).
## Coverage (Moderate Tasks)

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h^{\text{Boot}}$</th>
<th>$h^{\text{BExp}}$</th>
<th>$h^{\text{AVI}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>18.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>depots</td>
<td>60.3</td>
<td>32.7</td>
<td>54.7</td>
</tr>
<tr>
<td>grid</td>
<td>100.0</td>
<td>100.0</td>
<td>51.0</td>
</tr>
<tr>
<td>npuzzle</td>
<td>28.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>pipes-nt</td>
<td>57.8</td>
<td>68.4</td>
<td>50.2</td>
</tr>
<tr>
<td>rovers</td>
<td>48.2</td>
<td>21.8</td>
<td>45.0</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>33.3</td>
<td>70.7</td>
<td>67.3</td>
</tr>
<tr>
<td>storage</td>
<td>89.0</td>
<td>57.5</td>
<td>69.5</td>
</tr>
<tr>
<td>transport</td>
<td>100.0</td>
<td>100.0</td>
<td>87.5</td>
</tr>
<tr>
<td>visitall</td>
<td>55.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
# Coverage (Moderate Tasks)

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h^{Boot}$</th>
<th>$h^{BExp}$</th>
<th>$h^{AVI}$</th>
<th>$h^{SL}$</th>
<th>$h^{HGN}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>18.0</td>
<td>0.0</td>
<td>0.0</td>
<td>80.4</td>
<td><strong>100.0</strong></td>
</tr>
<tr>
<td>depots</td>
<td>60.3</td>
<td>32.7</td>
<td>54.7</td>
<td><strong>90.3</strong></td>
<td>0.0</td>
</tr>
<tr>
<td>grid</td>
<td><strong>100.0</strong></td>
<td><strong>100.0</strong></td>
<td>51.0</td>
<td>93.0</td>
<td>0.0</td>
</tr>
<tr>
<td>npuzzle</td>
<td>28.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>pipes-nt</td>
<td>57.8</td>
<td>68.4</td>
<td>50.2</td>
<td><strong>92.2</strong></td>
<td>7.6</td>
</tr>
<tr>
<td>rovers</td>
<td><strong>48.2</strong></td>
<td>21.8</td>
<td>45.0</td>
<td>26.0</td>
<td>14.0</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>33.3</td>
<td>70.7</td>
<td>67.3</td>
<td><strong>82.7</strong></td>
<td>11.0</td>
</tr>
<tr>
<td>storage</td>
<td><strong>89.0</strong></td>
<td>57.5</td>
<td>69.5</td>
<td>24.5</td>
<td>0.0</td>
</tr>
<tr>
<td>transport</td>
<td><strong>100.0</strong></td>
<td><strong>100.0</strong></td>
<td>87.5</td>
<td>99.2</td>
<td><strong>94.7</strong></td>
</tr>
<tr>
<td>visitall</td>
<td>55.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td><strong>100.0</strong></td>
</tr>
</tbody>
</table>
## Coverage (Moderate Tasks)

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h^{Boot}$</th>
<th>$h^{BExp}$</th>
<th>$h^{AVI}$</th>
<th>$h^{SL}$</th>
<th>$h^{HGN}$</th>
<th>$h^{FF}$</th>
<th>LAMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>18.0</td>
<td>0.0</td>
<td>0.0</td>
<td>80.4</td>
<td>100.0</td>
<td>98.8</td>
<td>100.0</td>
</tr>
<tr>
<td>depots</td>
<td>60.3</td>
<td>32.7</td>
<td>54.7</td>
<td>90.3</td>
<td>0.0</td>
<td>98.0</td>
<td>100.0</td>
</tr>
<tr>
<td>grid</td>
<td>100.0</td>
<td>100.0</td>
<td>51.0</td>
<td>93.0</td>
<td>0.0</td>
<td>96.0</td>
<td>100.0</td>
</tr>
<tr>
<td>npuzzle</td>
<td>28.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.3</td>
<td>97.5</td>
<td>100.0</td>
</tr>
<tr>
<td>pipes-nt</td>
<td>57.8</td>
<td>68.4</td>
<td>50.2</td>
<td>92.2</td>
<td>7.6</td>
<td>82.4</td>
<td>99.4</td>
</tr>
<tr>
<td>rovers</td>
<td>48.2</td>
<td>21.8</td>
<td>45.0</td>
<td>26.0</td>
<td>14.0</td>
<td>84.2</td>
<td>100.0</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>33.3</td>
<td>70.7</td>
<td>67.3</td>
<td>82.7</td>
<td>11.0</td>
<td>98.3</td>
<td>100.0</td>
</tr>
<tr>
<td>storage</td>
<td>89.0</td>
<td>57.5</td>
<td>69.5</td>
<td>24.5</td>
<td>0.0</td>
<td>48.0</td>
<td>38.5</td>
</tr>
<tr>
<td>transport</td>
<td>100.0</td>
<td>100.0</td>
<td>87.5</td>
<td>99.2</td>
<td>94.7</td>
<td>98.5</td>
<td>100.0</td>
</tr>
<tr>
<td>visitall</td>
<td>55.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
<td>93.3</td>
<td>100.0</td>
</tr>
</tbody>
</table>
## Coverage (Hard Tasks)

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h^{Boot}$</th>
<th>$h^{BExp}$</th>
<th>$h^{AVI}$</th>
<th>$h^{SL}$</th>
<th>$h^{HGN}$</th>
<th>$h^{FF}$</th>
<th>LAMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>50.0</td>
<td>61.6</td>
<td>96.8</td>
</tr>
<tr>
<td>depots</td>
<td>8.3</td>
<td>4.3</td>
<td>12.9</td>
<td>35.4</td>
<td>0.0</td>
<td>36.0</td>
<td>82.6</td>
</tr>
<tr>
<td>grid</td>
<td>87.8</td>
<td>95.0</td>
<td>70.5</td>
<td>60.2</td>
<td>0.0</td>
<td>53.2</td>
<td>100.0</td>
</tr>
<tr>
<td>npuzzle</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>33.2</td>
<td>86.5</td>
</tr>
<tr>
<td>pipes-nt</td>
<td>23.4</td>
<td>19.1</td>
<td>8.0</td>
<td>48.7</td>
<td>0.0</td>
<td>27.4</td>
<td>69.3</td>
</tr>
<tr>
<td>rovers</td>
<td>2.8</td>
<td>0.8</td>
<td>6.5</td>
<td>1.5</td>
<td>0.3</td>
<td>13.9</td>
<td>100.0</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>3.3</td>
<td>0.0</td>
<td>60.7</td>
<td>60.0</td>
<td>0.0</td>
<td>98.0</td>
<td>100.0</td>
</tr>
<tr>
<td>storage</td>
<td>27.2</td>
<td>13.2</td>
<td>15.8</td>
<td>0.0</td>
<td>0.0</td>
<td>13.8</td>
<td>11.5</td>
</tr>
<tr>
<td>transport</td>
<td>0.0</td>
<td>0.0</td>
<td>2.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>92.8</td>
</tr>
<tr>
<td>visitall</td>
<td>28.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
<td>74.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Expansions

(a) Grid

(b) Scanalyzer

(c) Storage

(d) Visitall
Conclusion

- three new per-instance RL heuristics

- large scale comparison to previous work
  - trained heuristics highly complementary
  - in general, model-based heuristics win
  - all our RL heuristics superior in Storage


References


