Learning Neural Search Policies for Classical Planning

Paweł Gomoluch\textsuperscript{1}, Dalal Alrajeh\textsuperscript{1}, Alessandra Russo\textsuperscript{1},
Antonio Bucchiarone\textsuperscript{2}

\textsuperscript{1}Imperial College London, \textsuperscript{2}Fondazione Bruno Kessler

ICAPS, October 2020
Setting

- Satisficing classical planning
- Forward search guided by an existing heuristic function (FF)
Learning search policies – problem statement

- Construct a planner which can adapt its search approach while solving a planning problem.
- Learn search policies tailored to specific problem distributions and performance objectives using RL.
Previous work – the discrete case

while not solved do

choose one of the available search routines

keep applying it to the open list for a fixed time $t_r$

end while

---

The discrete case – limitations

- The framework comes with a number of fixed parameters, e.g., $\epsilon$, $t_r$.
- The learner can choose between the routines but it can’t combine them, e.g., perform $\epsilon$-greedy local search.
Construct a parametrized search routine combining elements of various search techniques:

- For specific values of the parameters, the routine can assume any of the techniques on its own.
- For intermediate values of the parameters, it combines elements of various techniques.
Parametrized search routine – outline

- Interleave between $n_L$ global and $n_G$ local expansions.
- Optionally, randomize the order of node expansion or follow them with a number of random walks.
Overall, the search parameters include:

- $\epsilon$ – the probability of selecting a random node from the open list;
- $S$ – the number of expansions without progress necessary to trigger a random walk;
- $R$ – the number of random walks following a single node expansion;
- $L$ – the length of a random walk;
- $C$ – the number of node expansions in the global-local cycle;
- $c$ – the proportion of local search in the global-local cycle.
To capture information about the state of the search, we consider features such as, among others:

▶ the heuristic value of the initial state \( h(s_0) \);
▶ the lowest heuristic value encountered within the search \( h_{\text{min}} \);
▶ the time elapsed since the search started;
▶ the number of node expansions performed since the last change in the value of \( h_{\text{min}} \).
Two variants of the approach

- A feed-forward neural network mapping the search statistics to values of the search parameters.
- A *stateless* approach, in which we optimize the values of the search parameters directly (with no dependency on the search statistics).
The general idea behind *Evolution Strategies* is to introduce a distribution over possible solutions (sets of parameters).

At each iteration, the distribution is updated to maximize the likelihood of best-performing solutions.

---

CEM training – overview

initialize $\mu$ and $\Sigma$

for $i = 1...u$ do
  $p_1...p_r \leftarrow \mathcal{P}$
  $\theta_1...\theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$
  for $j = 1...n$ do
    for $k = 1...r$ do
      run policy $\theta_j$ on $p_k$, record plan cost $c_{j,k}$
    end for
  end for
end for

$G_1...G_n \leftarrow$ compute IPC score for $\theta_1...\theta_n$

sort $\theta_1...\theta_n$ by scores $G_1...G_n$ (highest first)

$\mu \leftarrow (1 - \alpha)\mu + \alpha \cdot \text{mean}(\theta_1...\theta_m)$
\[\Sigma \leftarrow (1 - \alpha)\Sigma + \alpha \cdot \text{covariance}(\theta_1...\theta_m)\]

end for

return $\mu$
CEM training – overview

initialize \( \mu \) and \( \Sigma \)

for \( i = 1 \ldots u \) do

\[ p_1 \ldots p_r \leftarrow P \rightarrow \text{sample } r \text{ problems} \]

\[ \theta_1 \ldots \theta_n \leftarrow N(\mu, \Sigma) \rightarrow \text{sample } n \text{ policies} \]

for \( j = 1 \ldots n \) do

for \( k = 1 \ldots r \) do

run policy \( \theta_j \) on \( p_k \), record plan cost \( c_{j,k} \)

end for

end for

\[ G_1 \ldots G_n \leftarrow \text{compute IPC score for } \theta_1 \ldots \theta_n \]

sort \( \theta_1 \ldots \theta_n \) by scores \( G_1 \ldots G_n \) (highest first)

\[ \mu \leftarrow (1 - \alpha) \mu + \alpha \cdot \text{mean}(\theta_1 \ldots \theta_m) \]

\[ \Sigma \leftarrow (1 - \alpha) \Sigma + \alpha \cdot \text{covariance}(\theta_1 \ldots \theta_m) \]

end for

return \( \mu \)
CEM training – overview

initialize $\mu$ and $\Sigma$

for $i = 1\ldots u$ do
  $p_1\ldots p_r \leftarrow P$
  $\theta_1\ldots \theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$
  for $j = 1\ldots n$ do
    for $k = 1\ldots r$ do
      run policy $\theta_j$ on $p_k$, record plan cost $c_{j,k}$
    end for
  end for
end for

$G_1\ldots G_n \leftarrow$ compute IPC score for $\theta_1\ldots \theta_n$

sort $\theta_1\ldots \theta_n$ by scores $G_1\ldots G_n$ (highest first)

$\mu \leftarrow (1 - \alpha)\mu + \alpha \cdot \text{mean}(\theta_1\ldots \theta_m)$

$\Sigma \leftarrow (1 - \alpha)\Sigma + \alpha \cdot \text{covariance}(\theta_1\ldots \theta_m)$

end for

return $\mu$
CEM training – overview

initialize $\mu$ and $\Sigma$

for $i = 1...u$ do
  $p_1...p_r \leftarrow P$ \hspace{1cm} ▷ sample $r$ problems
  $\theta_1...\theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$ \hspace{1cm} ▷ sample $n$ policies
  for $j = 1...n$ do
    for $k = 1...r$ do
      run policy $\theta_j$ on $p_k$, record plan cost $c_{j,k}$
    end for
  end for
end for

$G_1...G_n \leftarrow$ compute IPC score for $\theta_1...\theta_n$

sort $\theta_1...\theta_n$ by scores $G_1...G_n$ (highest first)

$\mu \leftarrow (1 - \alpha)\mu + \alpha \cdot \text{mean}(\theta_1...\theta_m)$

$\Sigma \leftarrow (1 - \alpha)\Sigma + \alpha \cdot \text{covariance}(\theta_1...\theta_m)$

end for

return $\mu$
CEM training – overview

initialize $\mu$ and $\Sigma$

for $i = 1...u$ do
  $p_1...p_r \leftarrow \mathcal{P}$ \hspace{1cm} ▶ sample $r$ problems
  $\theta_1...\theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$ \hspace{1cm} ▶ sample $n$ policies
  for $j = 1...n$ do
    for $k = 1...r$ do
      run policy $\theta_j$ on $p_k$, record plan cost $c_{j,k}$
    end for
  end for
end for

$G_1...G_n \leftarrow$ compute IPC score for $\theta_1...\theta_n$

sort $\theta_1...\theta_n$ by scores $G_1...G_n$ (highest first)

$\mu \leftarrow (1 - \alpha)\mu + \alpha \cdot \text{mean}(\theta_1...\theta_m)$
$\Sigma \leftarrow (1 - \alpha)\Sigma + \alpha \cdot \text{covariance}(\theta_1...\theta_m)$

end for

return $\mu$
CEM training – overview

initialize $\mu$ and $\Sigma$

for $i = 1...u$ do
  $p_1...p_r \leftarrow P$
  $\theta_1...\theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$
  for $j = 1...n$ do
    for $k = 1...r$ do
      run policy $\theta_j$ on $p_k$, record plan cost $c_{j,k}$
    end for
  end for
end for

$G_1...G_n \leftarrow$ compute IPC score for $\theta_1...\theta_n$

sort $\theta_1...\theta_n$ by scores $G_1...G_n$ (highest first)

$\mu \leftarrow (1 - \alpha)\mu + \alpha \cdot \text{mean}(\theta_1...\theta_m)$
$\Sigma \leftarrow (1 - \alpha)\Sigma + \alpha \cdot \text{covariance}(\theta_1...\theta_m)$

end for

return $\mu$
CEM training – overview

initialize $\mu$ and $\Sigma$

for $i = 1...u$ do
    $p_1...p_r \leftarrow P$  \hspace{1cm} ▷ sample $r$ problems
    $\theta_1...\theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$  \hspace{1cm} ▷ sample $n$ policies

for $j = 1...n$ do
    for $k = 1...r$ do
        for $k = 1...r$ do
            run policy $\theta_j$ on $p_k$, record plan cost $c_{j,k}$
        end for
    end for
end for

$G_1...G_n \leftarrow \text{compute IPC score for } \theta_1...\theta_n$

sort $\theta_1...\theta_n$ by scores $G_1...G_n$ (highest first)

$\mu \leftarrow (1 - \alpha)\mu + \alpha \cdot \text{mean}(\theta_1...\theta_m)$

$\Sigma \leftarrow (1 - \alpha)\Sigma + \alpha \cdot \text{covariance}(\theta_1...\theta_m)$

end for

return $\mu$
Evaluation

- Five planning of the IPC learning track: *Transport*, *Parking*, *Elevators*, *No-mystery* and *Floortile*.
- Problem distributions matching the problem sets of IPC 2011 satisficing track.
- A timeout of 3 minutes.
Results – IPC scores

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>F</th>
<th>N</th>
<th>P</th>
<th>T</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBFS</td>
<td>14.67</td>
<td>2.24</td>
<td>8.18</td>
<td>9.24</td>
<td>2.6</td>
<td>36.93</td>
</tr>
<tr>
<td>$\epsilon$-greedy</td>
<td>13.07</td>
<td>2.64</td>
<td>8.93</td>
<td>7.44</td>
<td>2.7</td>
<td>34.78</td>
</tr>
<tr>
<td>RW</td>
<td>14.63</td>
<td>0.47</td>
<td>6.78</td>
<td>7.95</td>
<td>3.6</td>
<td>33.42</td>
</tr>
<tr>
<td>Local</td>
<td>15.97</td>
<td>1.91</td>
<td>7.15</td>
<td>11.85</td>
<td>4.48</td>
<td>41.36</td>
</tr>
<tr>
<td>Mixed</td>
<td>11.6</td>
<td>1.25</td>
<td>6.69</td>
<td>6.14</td>
<td>2.9</td>
<td>28.58</td>
</tr>
<tr>
<td>Opt</td>
<td>14.64</td>
<td>3.5</td>
<td>8.86</td>
<td>13.81</td>
<td>5.39</td>
<td>46.18</td>
</tr>
<tr>
<td>NSP</td>
<td>16.37</td>
<td>3.28</td>
<td>9.04</td>
<td>12.93</td>
<td>5.12</td>
<td>46.74</td>
</tr>
</tbody>
</table>

IPC scores for randomly generated test problems (average over 10 sets). Elevators (E), Floortile (F), No-mystery (N), Parking (P) and Transport (T).
Conclusion and future work

Contributions:

- Parametrized search routine combining elements of various search techniques.
- *Search policy* model, mapping the state of the search to values of the routine’s parameters.
- Evolutionary training scheme based on CEM.

Directions for future work:

- Extending the search routine with multiple open lists and novelty-based search.
- More complex representation of the planner’s state.