Learning Neural Search Policies for Classical Planning

Paweł Gomoluch[1], Dalal Alrajeh[1], Alessandra Russo[1], Antonio Bucchiarone[2]

[1]Imperial College London, [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[1]

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

[1] Pawel Gomoluch, Dalal Alrajeh, and Alessandra Russo. "Learning Classical Planning Strategies with Policy Gradient". In: *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2019*. 2019.

The discrete case – limitations

- The framework comes with a number of fixed parameters, e.g., ϵ, t_r.
- The learner can choose between the routines but it can't combine them, e.g., perform ϵ-greedy local search.

Solution – a continuous action space

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.

Search parameters – summary

Overall, the search parameters include:

- ϵ – 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.

Representation of the planner's state

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 $h_{\min}$;
- the time elapsed since the search started;
- the number of node expansions performed since the last change in the value of $h_{\min}$.

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).

Training – the *Cross-Entropy Method*[2]

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.

[2] Shie Mannor, Reuven Rubinstein, and Yohai Gat. "The Cross Entropy Method for Fast Policy Search". In: *ICML'03 Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. 2003.

CEM training – overview

```
initialize μ and Σ
for i = 1...u do
    p_1...p_r ← 𝒫                                        ▷ sample r problems
    θ_1...θ_n ← 𝒩(μ, Σ)                                  ▷ sample n policies
    for j = 1...n do
        for k = 1...r do
            run policy θ_j on p_k, record plan cost c_{j,k}
        end for
    end for
    G_1...G_n ← compute IPC score for θ_1...θ_n
    sort θ_1...θ_n by scores G_1...G_n (highest first)
    μ ← (1 − α)μ + α · mean(θ_1...θ_m)
    Σ ← (1 − α)Σ + α · covariance(θ_1...θ_m)
end for
return μ
```

CEM training – overview

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

CEM training – overview

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

CEM training – overview

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

CEM training – overview

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

CEM training – overview

initialize μ and Σ
for $i = 1...u$ **do**
    $p_1...p_r \leftarrow \mathcal{P}$ ▷ sample r problems
    $\theta_1...\theta_n \leftarrow \mathcal{N}(\mu, \Sigma)$ ▷ sample n policies
    for $j = 1...n$ **do**
        for $k = 1...r$ **do**
            run policy θ_j on p_k, record plan cost $c_{j,k}$
        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)
    $\mu \leftarrow (1-\alpha)\mu + \alpha \cdot \mathrm{mean}(\theta_1...\theta_m)$
    $\Sigma \leftarrow (1-\alpha)\Sigma + \alpha \cdot \mathrm{covariance}(\theta_1...\theta_m)$
end for
return μ

CEM training – overview

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

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

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

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.