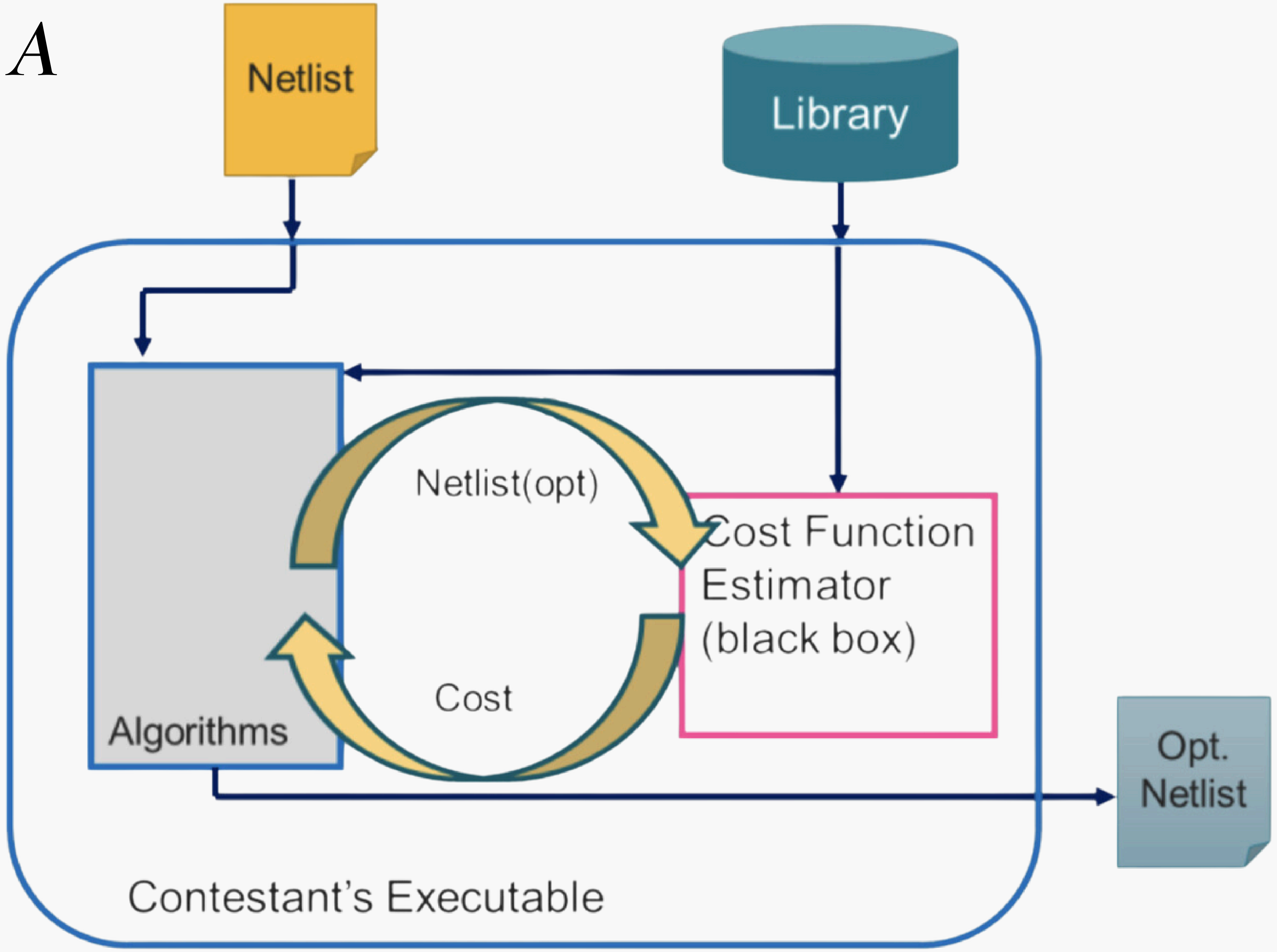
CAD Contest Problem A:

Reinforcement Logic Optimization for a General Cost Function

B10901027 楊竣凱
B10901098 蔡承恩
B10901106 田庭瑄

Introduction

Problem A



Goal:

Develop a program that interacts with a cost function estimator (black box) and learns to perform optimizations to minimize the cost of the circuit.

Introduction

Provided Files:

1. **Netlist (.v)**:

   A file that contains the target circuit we need to optimize.

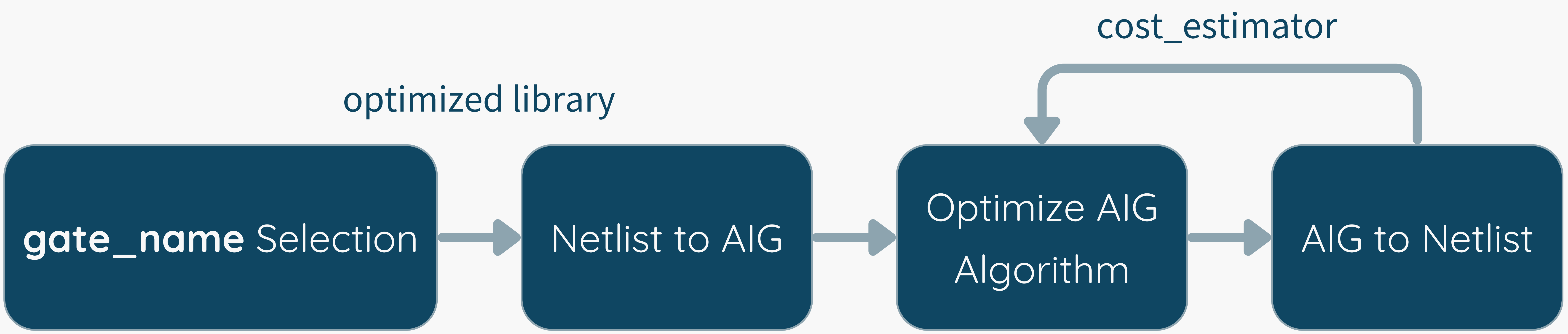
2. **Library (.json)**:

   A file with a list of gates and their corresponding costs, which we can use for mapping.

3. **Cost Estimator**:

   An executable tool that calculates the cost of the current netlist configuration.

Work Flow



Find the **gate_name** with lowest cost estimated by cost_estimator of each gate type

- Use abc commands to manipulate AIG
- Algorithm for optimization
    - Greedy Algorithm
    - Simulated Annealing
    - Fast Simulated Annealing
    - Reinforcement Learning

gate_name Selection

```
"cells" : [
    {
        "cell_name" : "and_1" ,
        "cell_type" : "and" ,
        "data_1_f" : "4.106773" ,
        "data_2_f" : "0.083529" ,
        "data_3_i" : "2" ,
        "data_4_f" : "0.001751" ,
        "data_5_f" : "0.008245" ,
        "data_6_f" : "6.256000" ,
        "data_7_f" : "0.130930"
    } ,
```

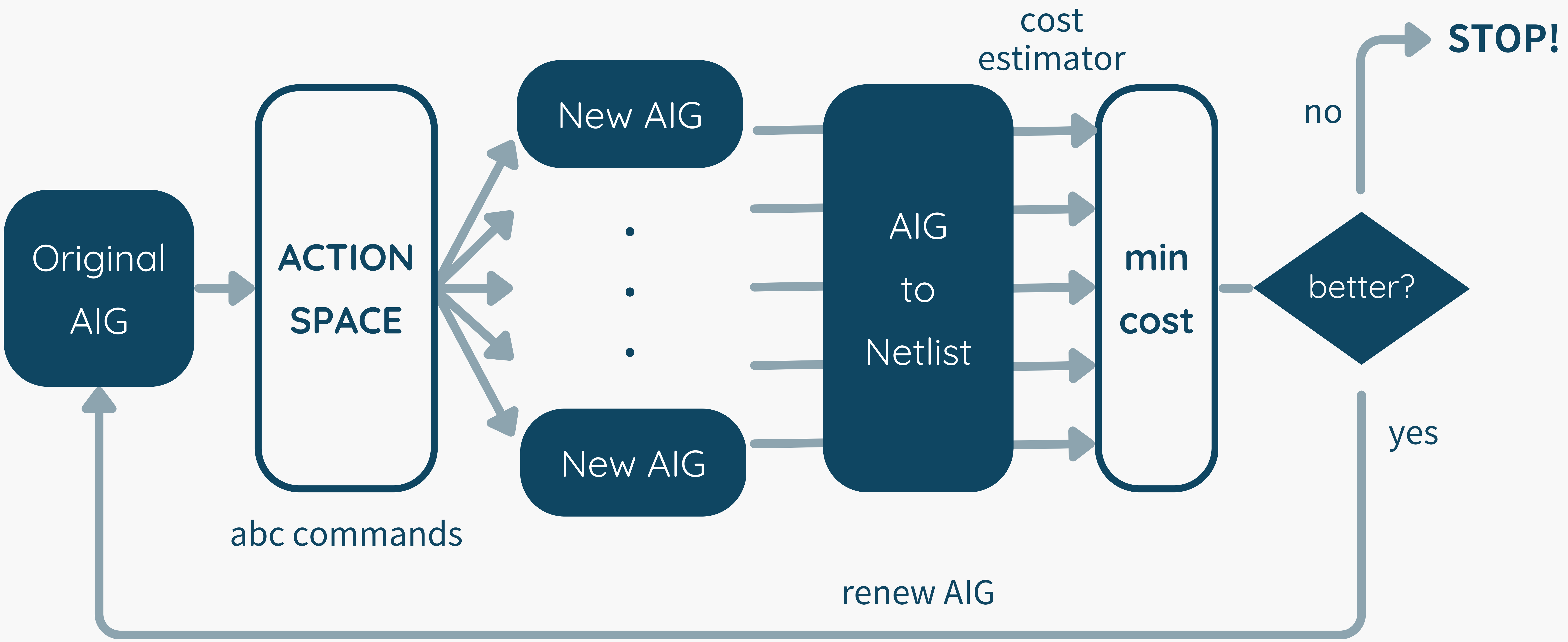
Given Library

cost_estimator →

```
library(demo) {
    cell(and_6) {
        cost: 0.050969;
        pin(Y) {
            direction: output;
            function: "A*B";
        }
        pin(A) { direction: input; }
        pin(B) { direction: input; }
    }
```

Optimized Library for mapping
AIG to Netlist

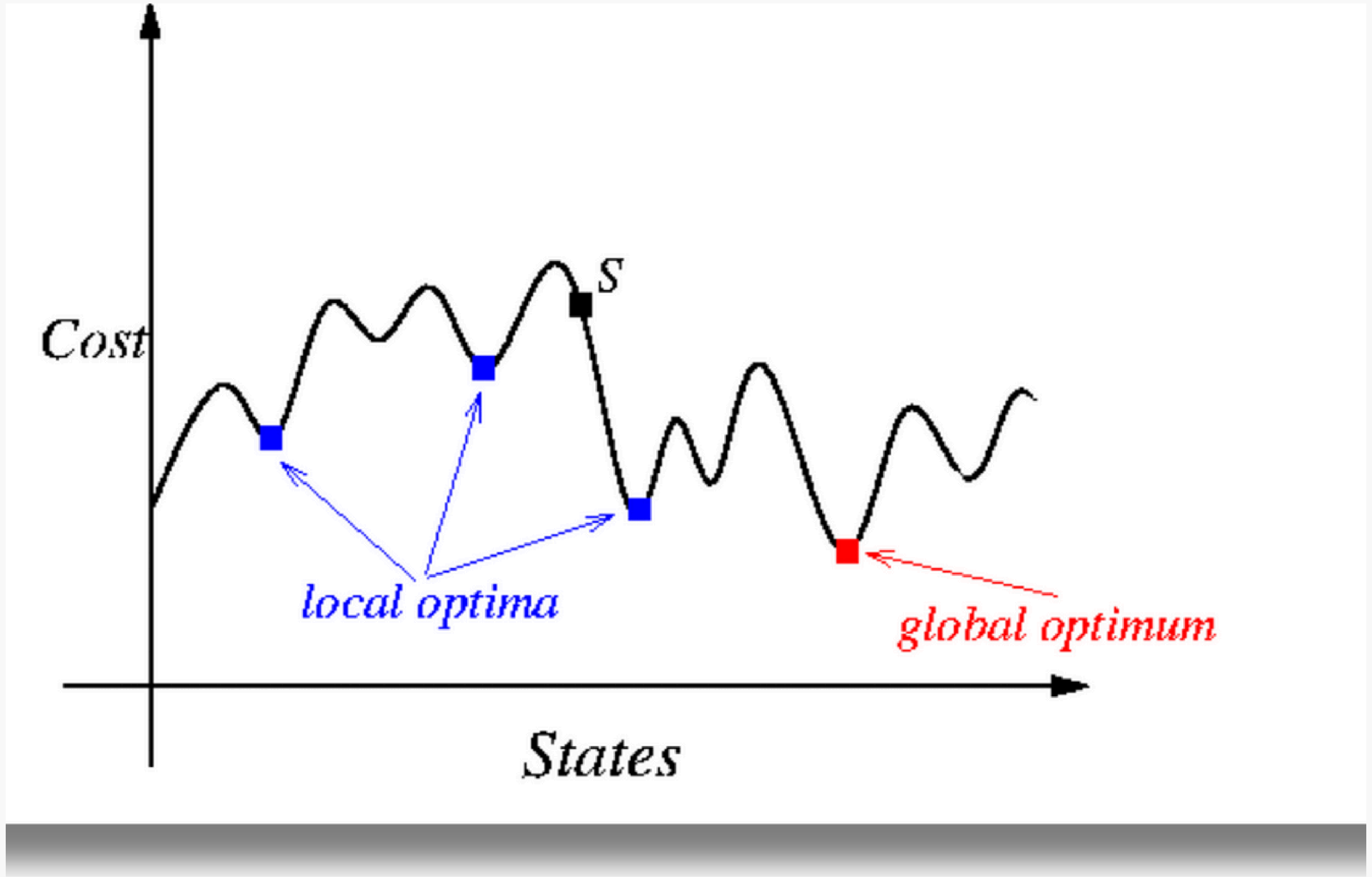
Algorithm - Greedy



Algorithm - Simulated Annealing

```
begin
Get an initial solution S;
Get an initial temperature T > 0;
while not yet "frozen" do
  for 1 ≤ i ≤ P do
      Pick a random neighbor S' of S;
      Δ ← cost(S') - cost(S);
      /* downhill move */
      if  Δ ≤ 0 then S ← S'
      /* uphill move */
      if  Δ > 0 then S ← S' with probability e^(-Δ/T) ;
  T ← rT;  /* reduce temperature */
return S
end
```

$$Prob(S \rightarrow S') = \begin{cases} 1 & \text{if } \Delta C \leq 0 \quad /* \text{ “}down-hill\text{'' } moves */ \\ e^{-\frac{\Delta C}{T}} & \text{if } \Delta C > 0 \quad /* \text{ “}up-hill\text{'' } moves */ \end{cases}$$

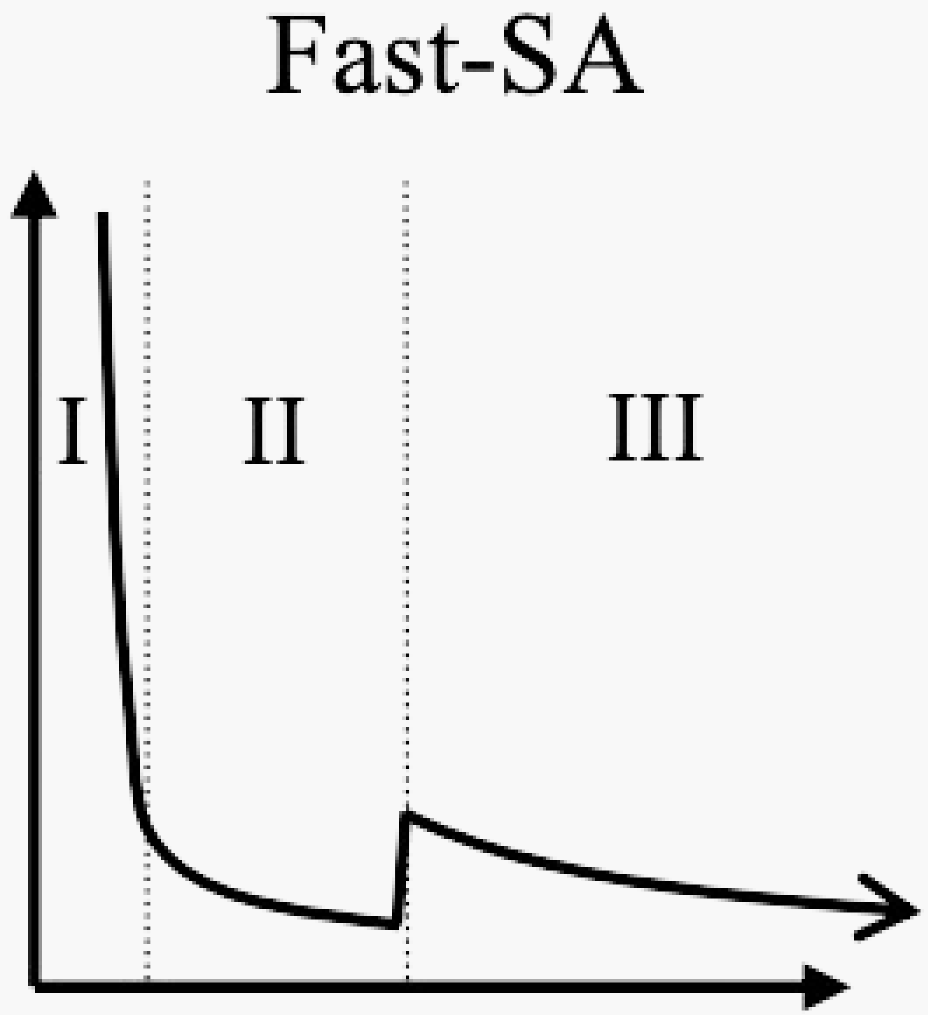


Algorithm - Fast Simulated Annealing

3-Phases:

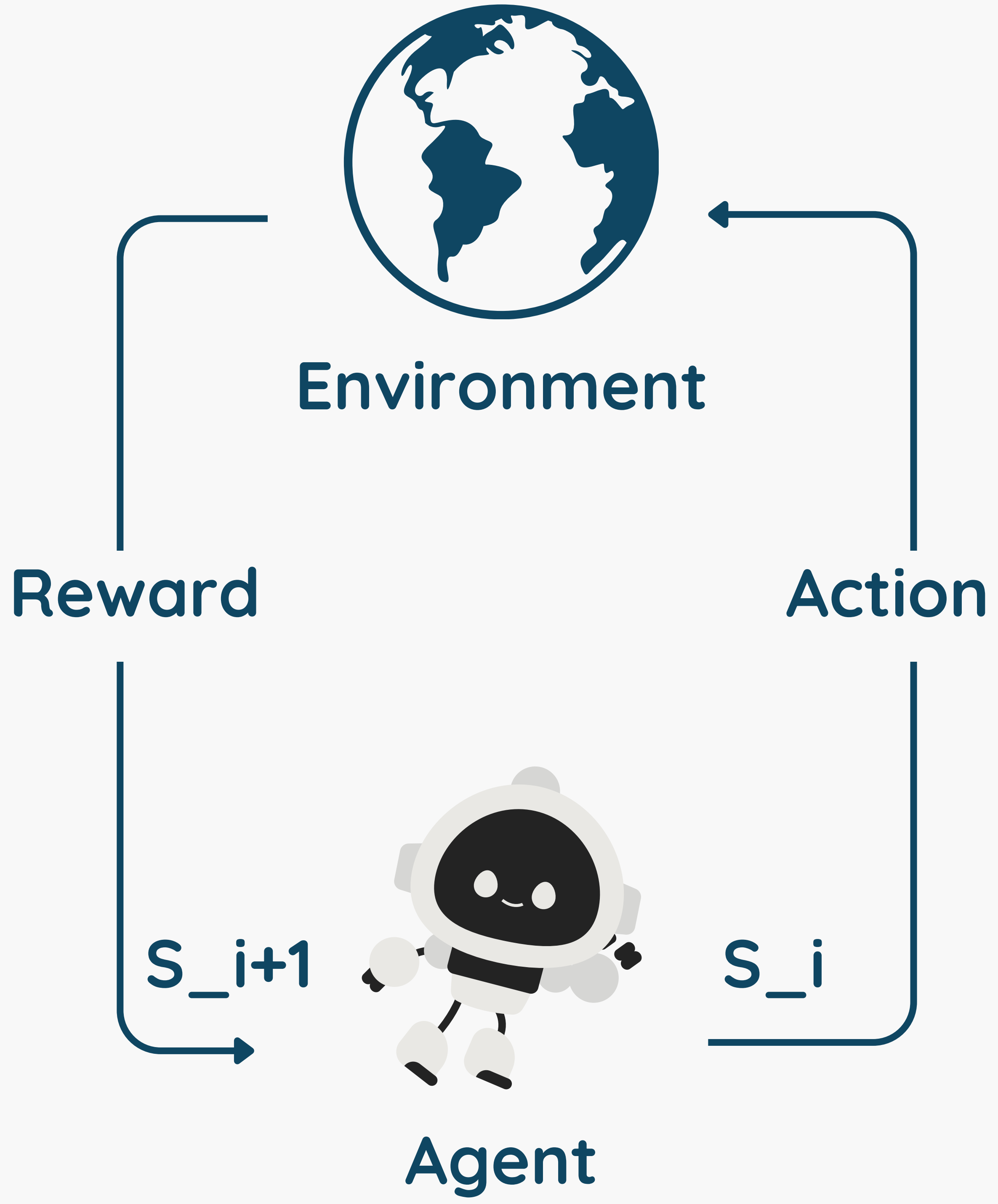
1. **Exploration:** Very High T
2. **Pseudo-greedy searches:** Very low T
3. **Improving:** Back to higher T

$$T_n = \begin{cases} \frac{\Delta_{avg}}{\ln P} & n = 1 \\ \frac{T_1 \langle \Delta_{cost} \rangle}{nc} & 2 \leq n \leq k \\ \frac{T_1 \langle \Delta_{cost} \rangle}{n} & n > k. \end{cases}$$



Tung-Chieh Chen and Yao-Wen Chang. 2005. Modern floorplanning based on **fast simulated annealing**. In Proceedings of the 2005 international symposium on Physical design (ISPD '05).

Algorithm - Reinforcement Learning



Foundation:

Markov decision process (MDP)
(S, A, π, R)

- S: state
- A: action
- π: policy, state-action transition distribution
- R: reward (or expected reward)

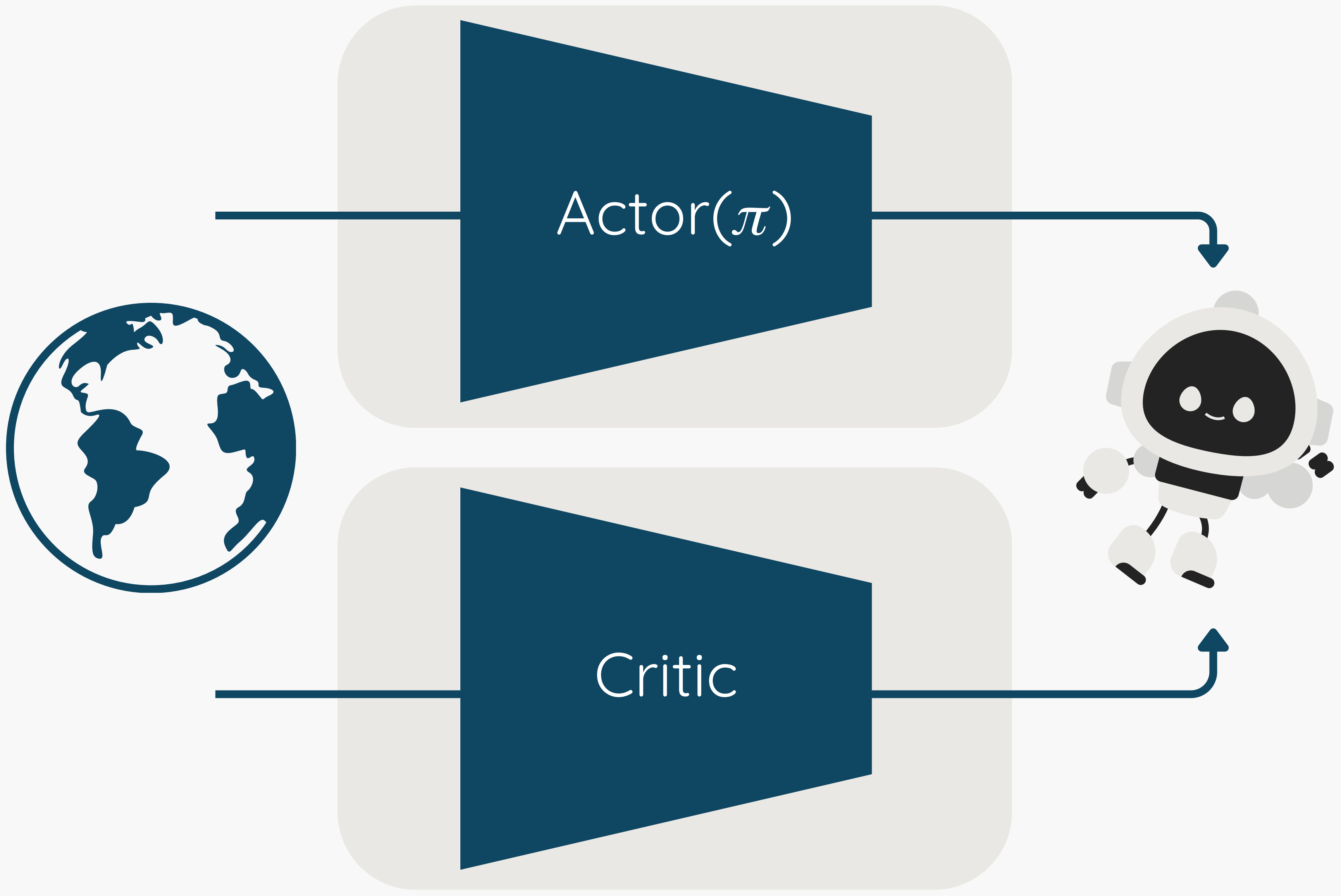
Q-learning:

- V: (State) Value function (Estimated Discount Return)
- Q: Q-function, State-action value function

Algorithm - Reinforcement Learning

Advantage Actor Critic (A2C)

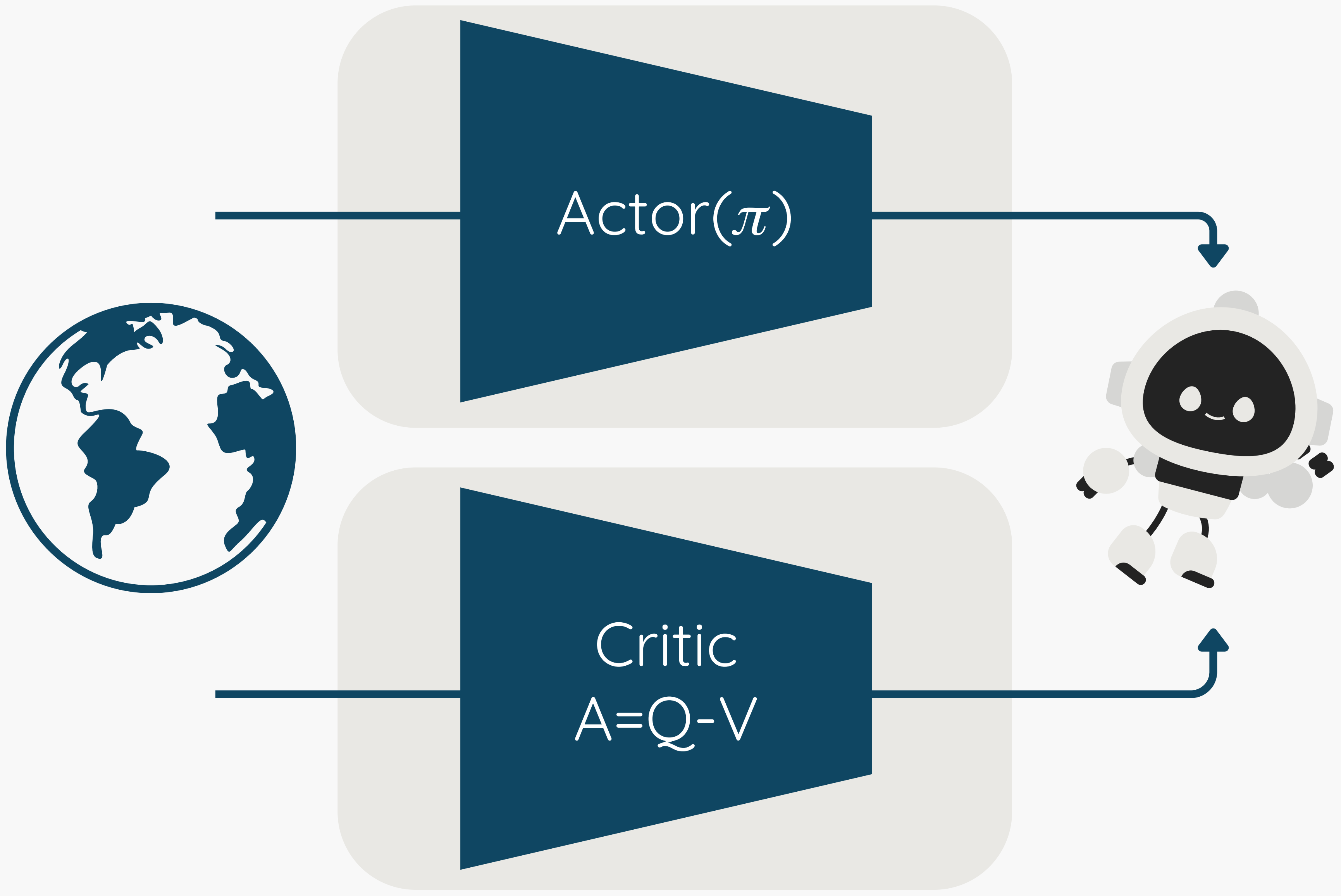
- Actor Critic



Algorithm - Reinforcement Learning

Advantage Actor Critic (A2C)

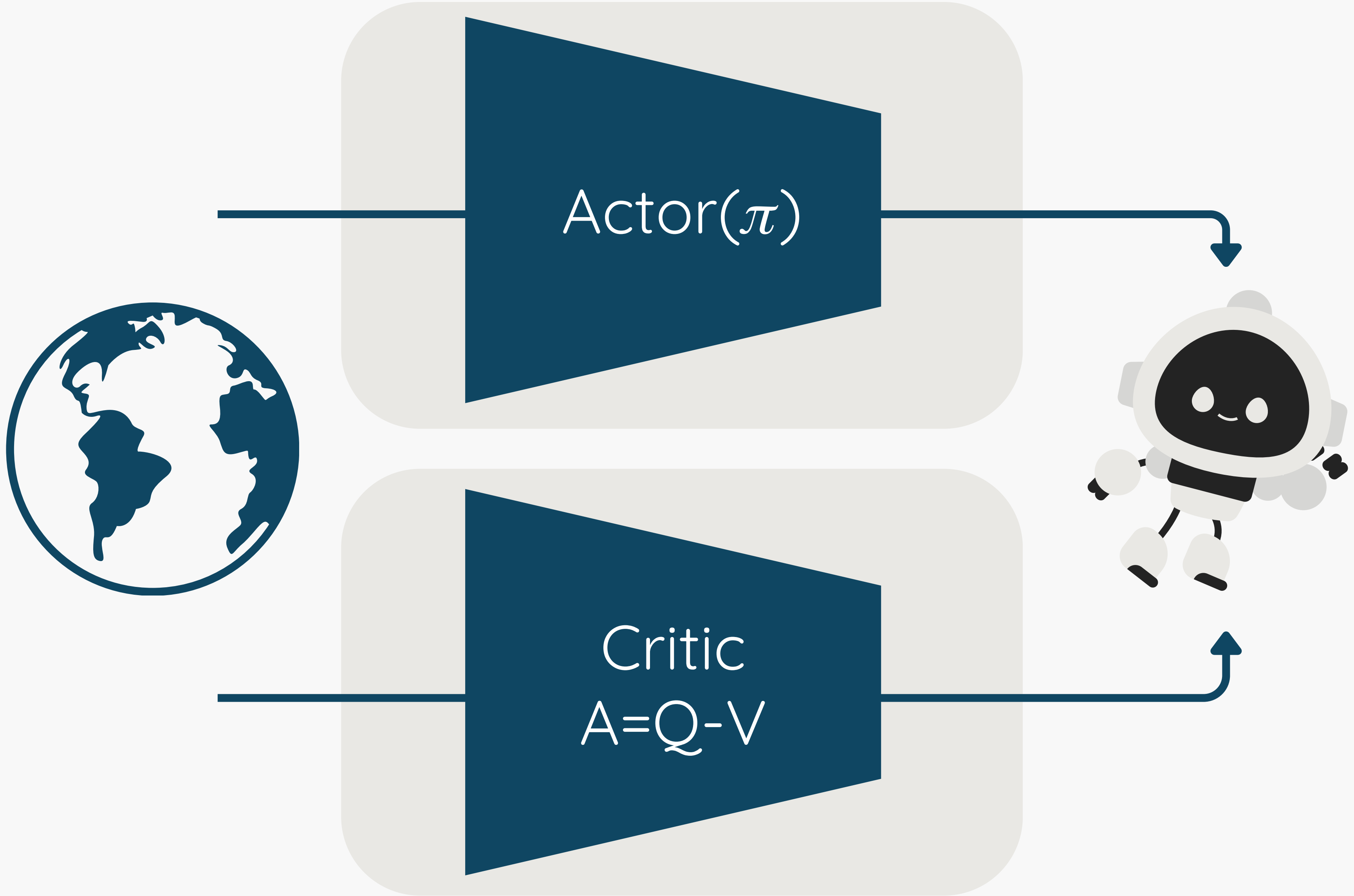
- Advantage Actor Critic



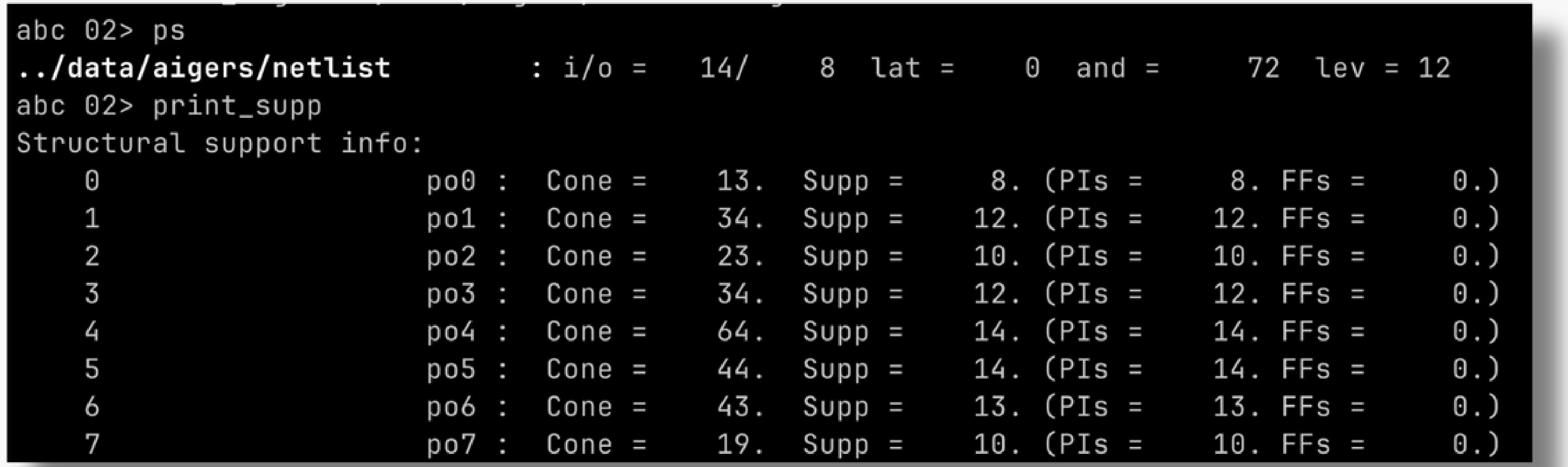
Algorithm - Reinforcement Learning

Advantage Actor Critic (A2C)

- Advantage Actor Critic



State Definition:



```
abc 02> ps
../data/aigers/netlist        : i/o =    14/    8  lat =    0  and =     72  lev = 12
abc 02> print_supp
Structural support info:
    0                     po0 :  Cone =     13.  Supp =     8. (PIs =     8. FFs =     0.)
    1                     po1 :  Cone =     34.  Supp =    12. (PIs =    12. FFs =     0.)
    2                     po2 :  Cone =     23.  Supp =    10. (PIs =    10. FFs =     0.)
    3                     po3 :  Cone =     34.  Supp =    12. (PIs =    12. FFs =     0.)
    4                     po4 :  Cone =     64.  Supp =    14. (PIs =    14. FFs =     0.)
    5                     po5 :  Cone =     44.  Supp =    14. (PIs =    14. FFs =     0.)
    6                     po6 :  Cone =     43.  Supp =    13. (PIs =    13. FFs =     0.)
    7                     po7 :  Cone =     19.  Supp =    10. (PIs =    10. FFs =     0.)
```

Action Definition:

3 consecutive actions composed of

- 26 proper commands from **abc.rc**
- None (no action)

→ total 27^3 combinations

Result & Comparison

cost	Greedy	SA	FSA	RL
test_case1	2.4966	2.4368	2.4368	2.4277
test_case2	40.8392	41.6155	42.7058	40.0006
test_case3	53.0899	53.5426	53.5162	52.6597
test_case4	143.7540	137.0197	135.9404	140.2703
test_case5	948.3810	938.5901	951.5275	935.5502

RL Implemtation Insight

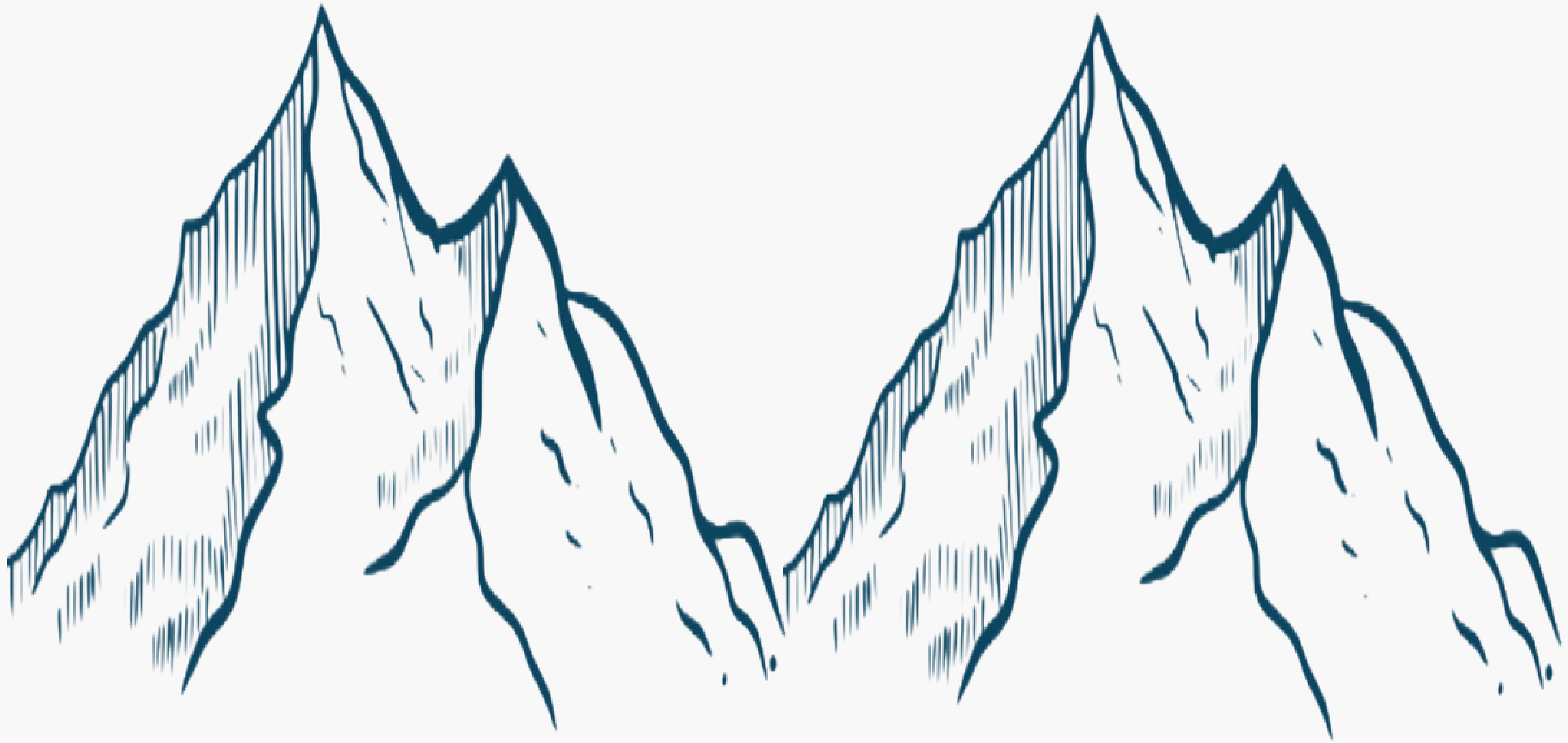
Perform better on Medium-Horizon task

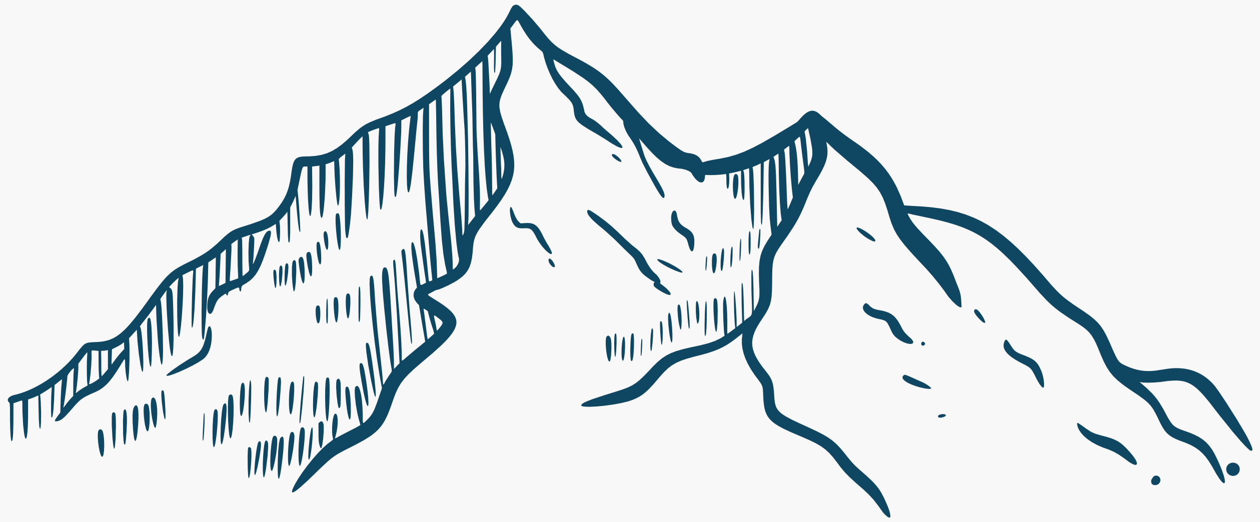
- It explores the state space but also avoids conducting “learned” repetitive actions.
- Crtic network can memorize long-term outcomes of actions.

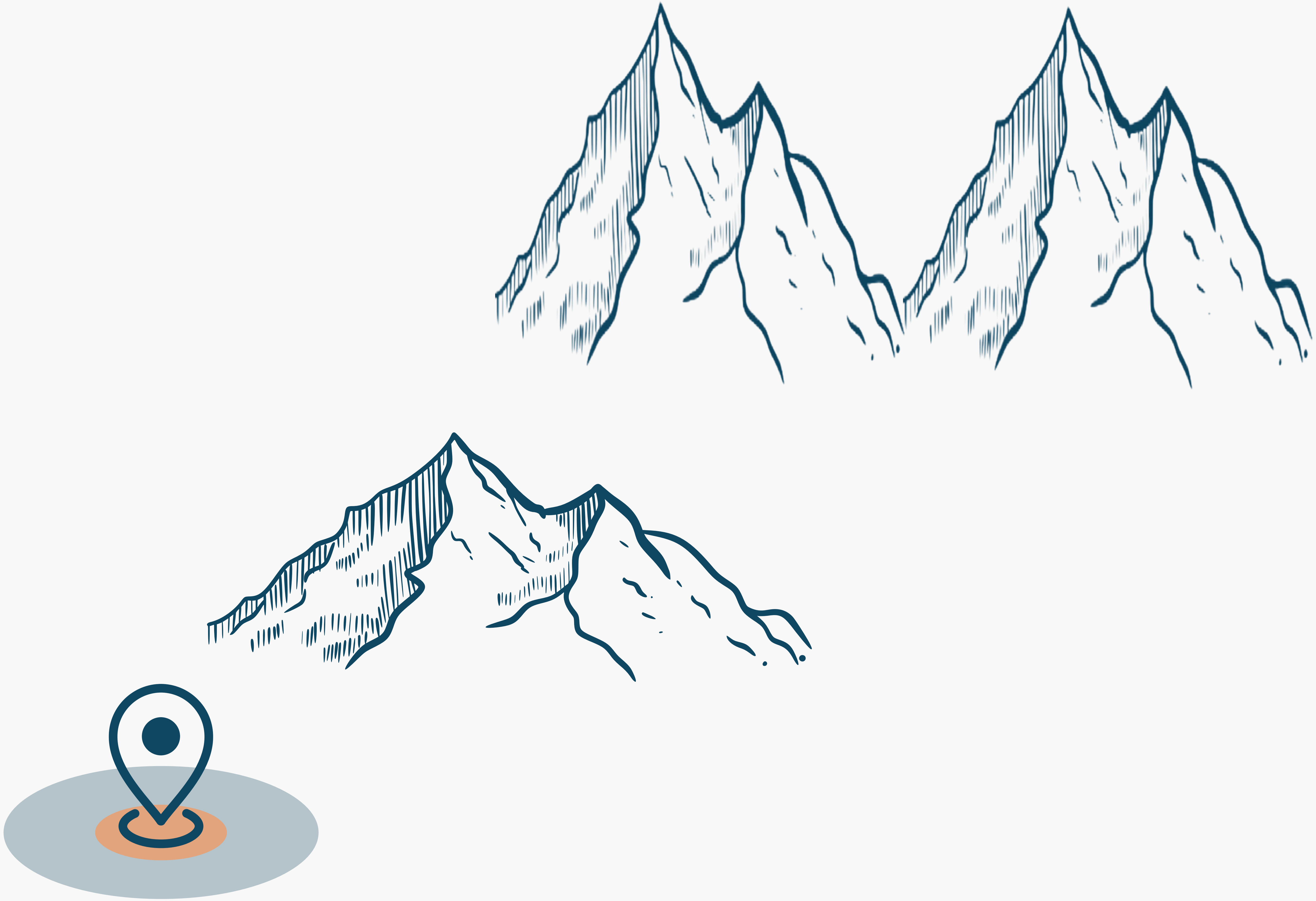
Very small learning rate & Early-stop episodes

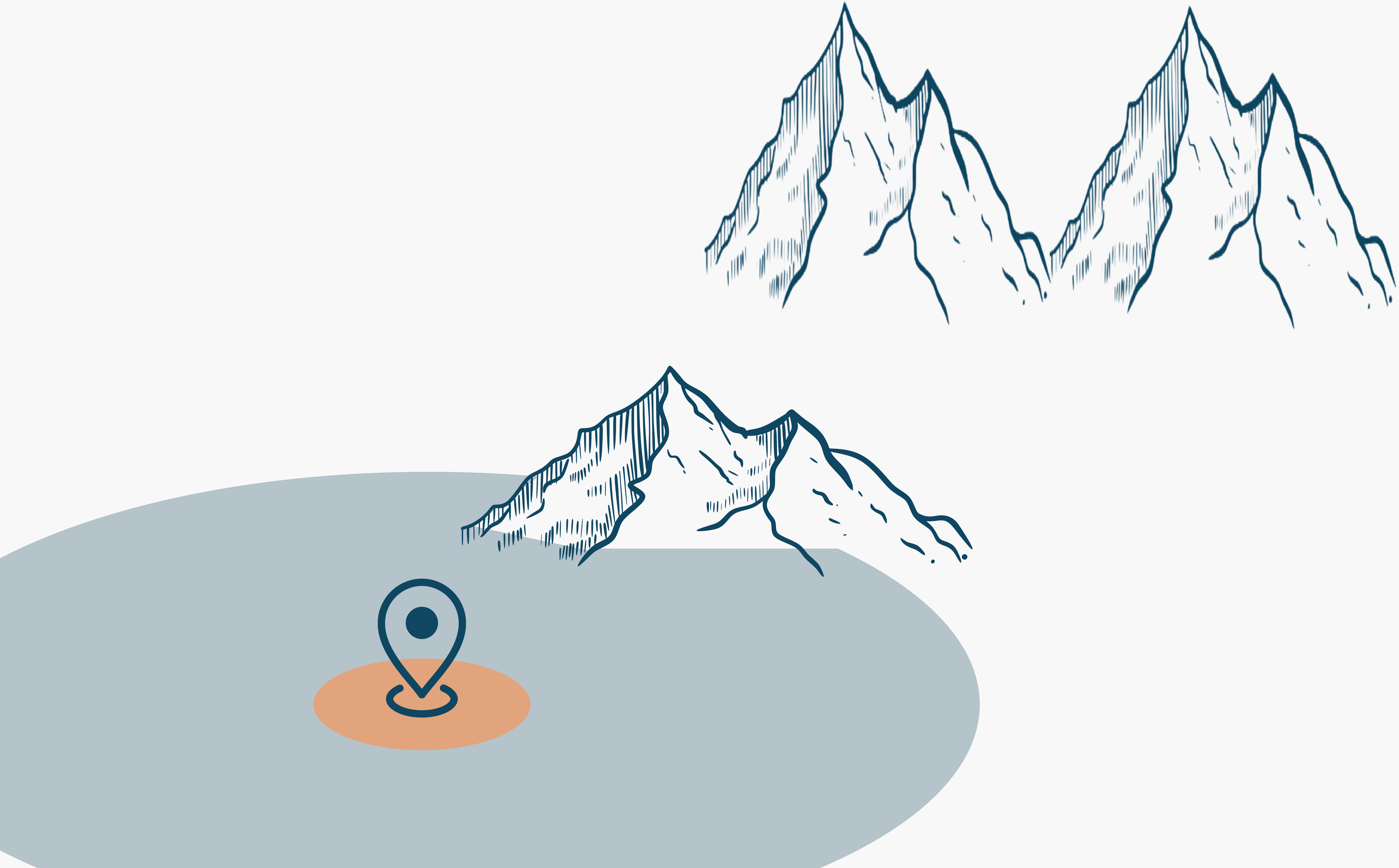
- RL should have a long enough exploration phase and not stuck at any specific state.
- It should remain some randomness to explore but with the ability to pass through explored states.

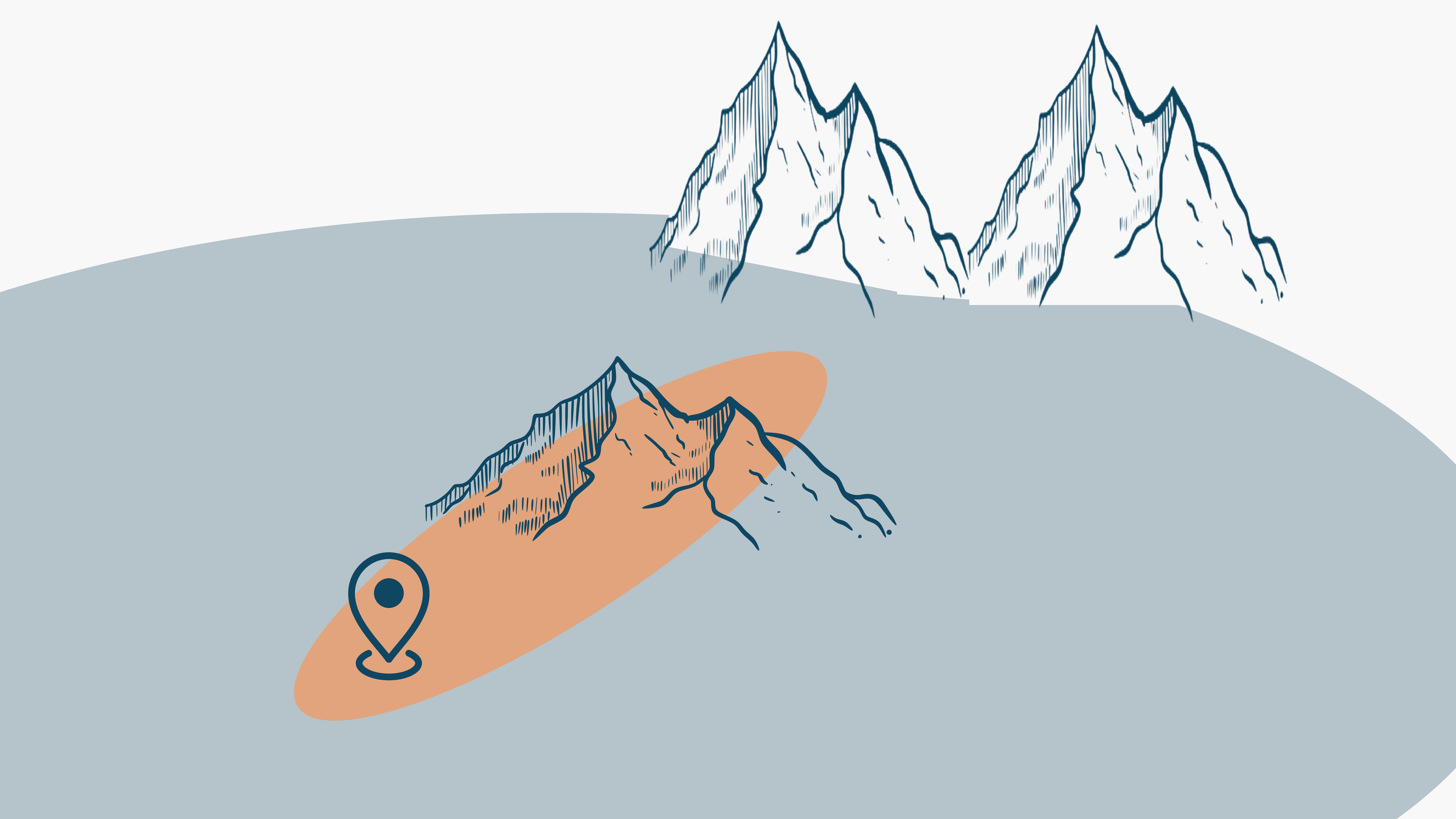
A pruned version of S.A.?

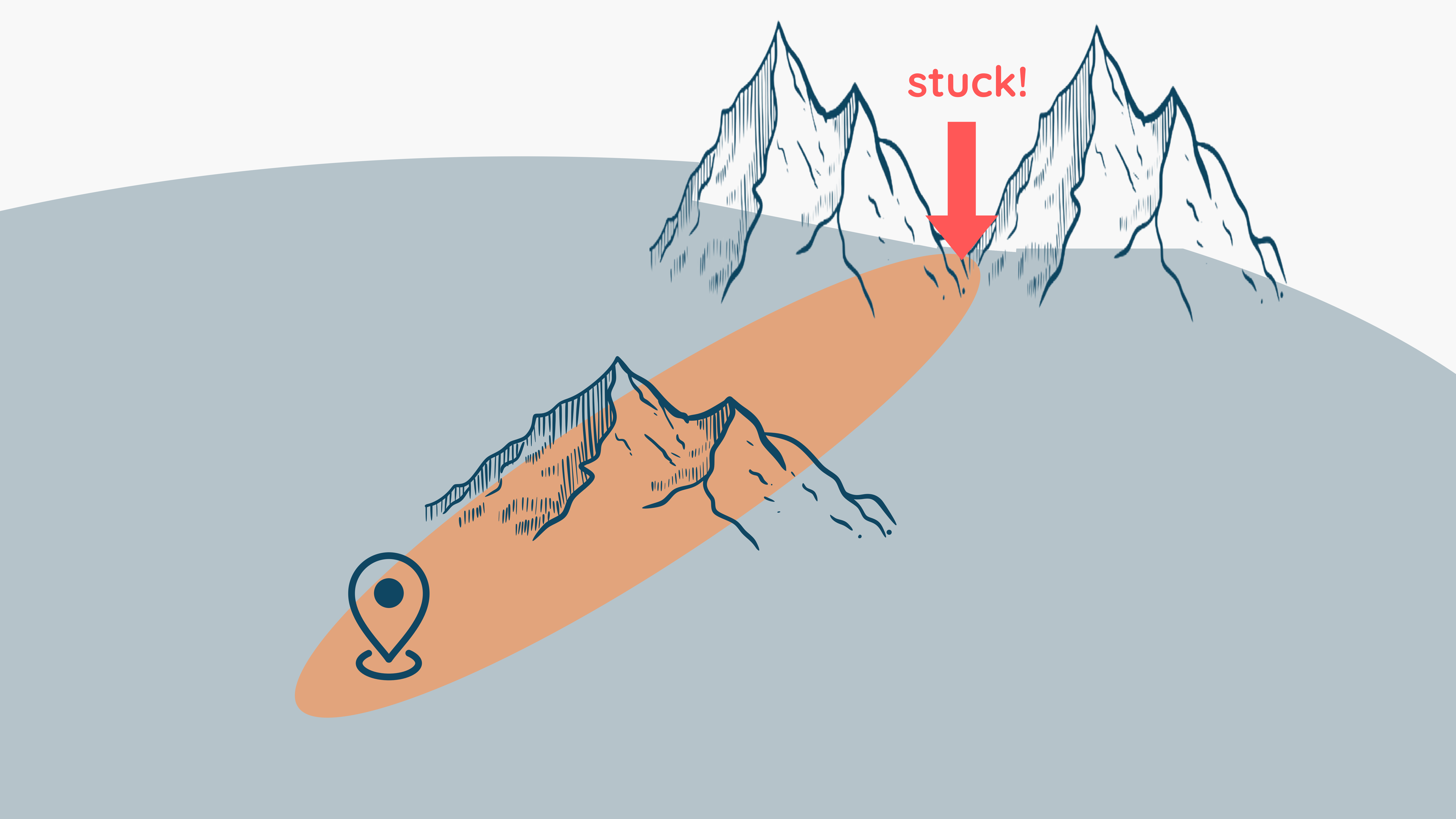











stuck!

RL Limitation

Not Markov decision process (MDP)

- underrepresented states: **no computationally feasible** representation can represent a unique graph with a large number of nodes

Sampling efficiency

- RL requires a large number of interactions with the environment, which is impractical for large task

Reward Function Engineering

- RL problems often require sophisticated reward function design, making it ungeneralizable

Thank you