DeepGo: Predictive Directed Greybox Fuzzing

Reporter: Huiling Chen

Peihong Lin, Pengfei Wang✉, Xu Zhou, Wei Xie, Gen Zhang, Kai Lu✉

National University of Defense Technology

Contact: phlin22@nudt.edu.cn

- PART 1 Background and Motivation
- PART 2 Design
- PART 3 Evaluations
- PART 4 Conclusion

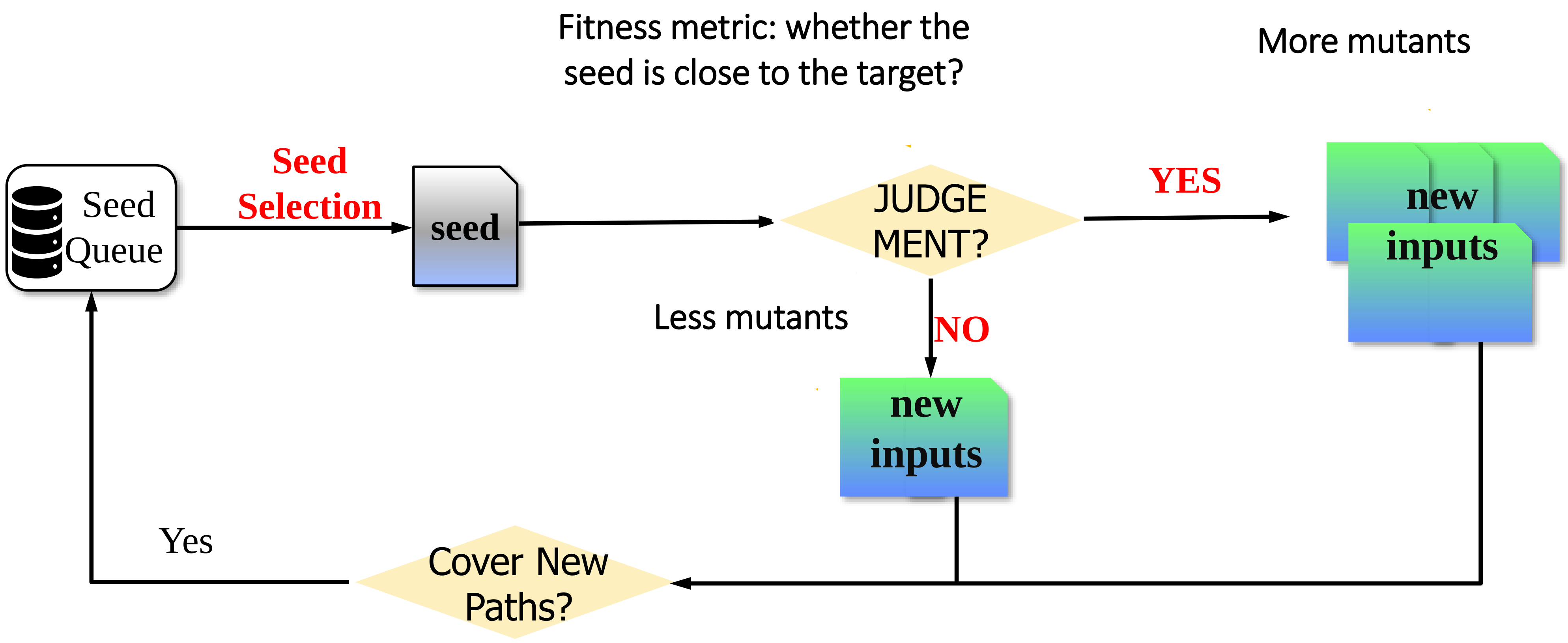
- PART 1 Background and Motivation
- PART 2 Design
- PART 3 Evaluations
- PART 4 Conclusion

1. Background and Motivation

- **Fuzzing**
  - Effective approach to discovering vulnerabilities
  - e.g., AFL, Google’s OSS Fuzz
- **Directed Greybox Fuzzing (DGF)**
  - Designed technique for testing the given target code locations
  - Patch testing, bug reproduction, potential buggy code verification

1. Background and Motivation

- **Directed Greybox Fuzzing (DGF)**



1. Background and Motivation

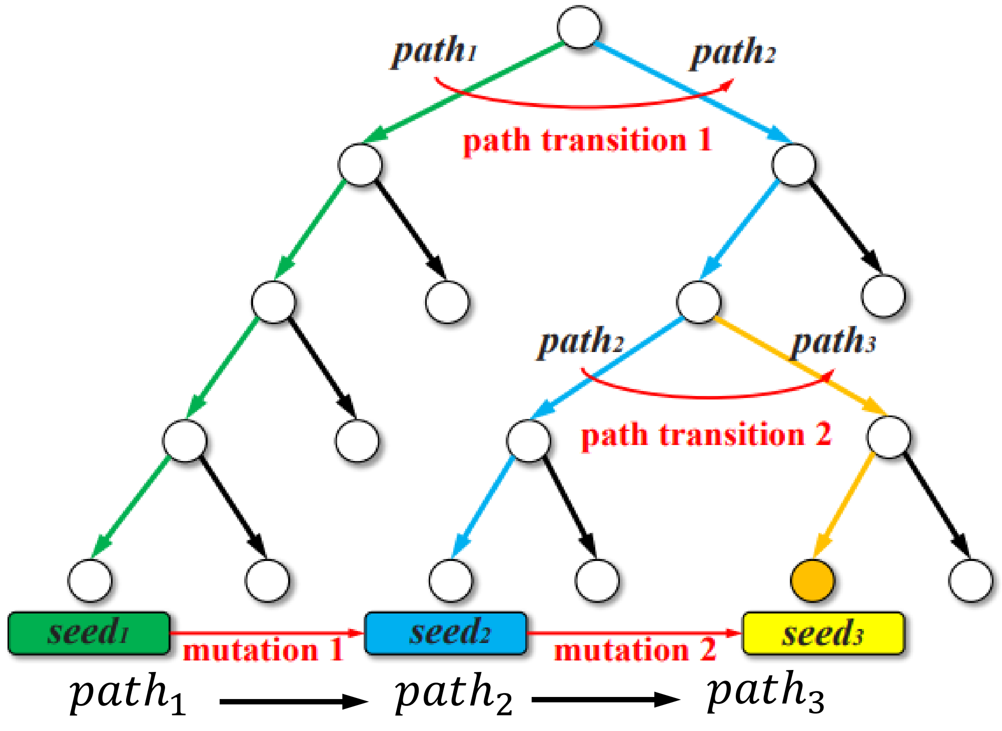
- **State-of-the-art DGF techniques**
  - The state-of-the-art DGF works leverage heuristic methods to optimize fitness metrics or exclude the irrelevant code locations.
    - e.g., BEACON (path pruning), CAFL and WindRanger (data condition)

- **However**
  - Heuristic methods **lack foresight** on paths that have not been exercised yet
  - Hard-to-execute paths with complex constraints would hinder DGF

- **For example**
  - Using BB distance, seeds with shorter distances are prioritized
  - Complex constraints along seeds' paths will hinder fuzzer from reaching targets

1. Background and Motivation

- **Our goal**
  - **Path Transition Model**.
    - Model DGF as a process of reaching the target site through specific path transition sequences.
  - Design a predictive directed greybox fuzzer to **predict the path transitions**.
  - Intelligently generate the optimal and viable path to the target site.



1. Background and Motivation

- **Challenges**
  - ***Challenge 1:*** How to predict path transitions that have not been taken?
  - ***Challenge 2:*** How to determine the optimal path among large numbers of path transitions?
  - ***Challenge 3:*** How to exercise the optimal path transition sequences by optimizing the fuzzing strategies?

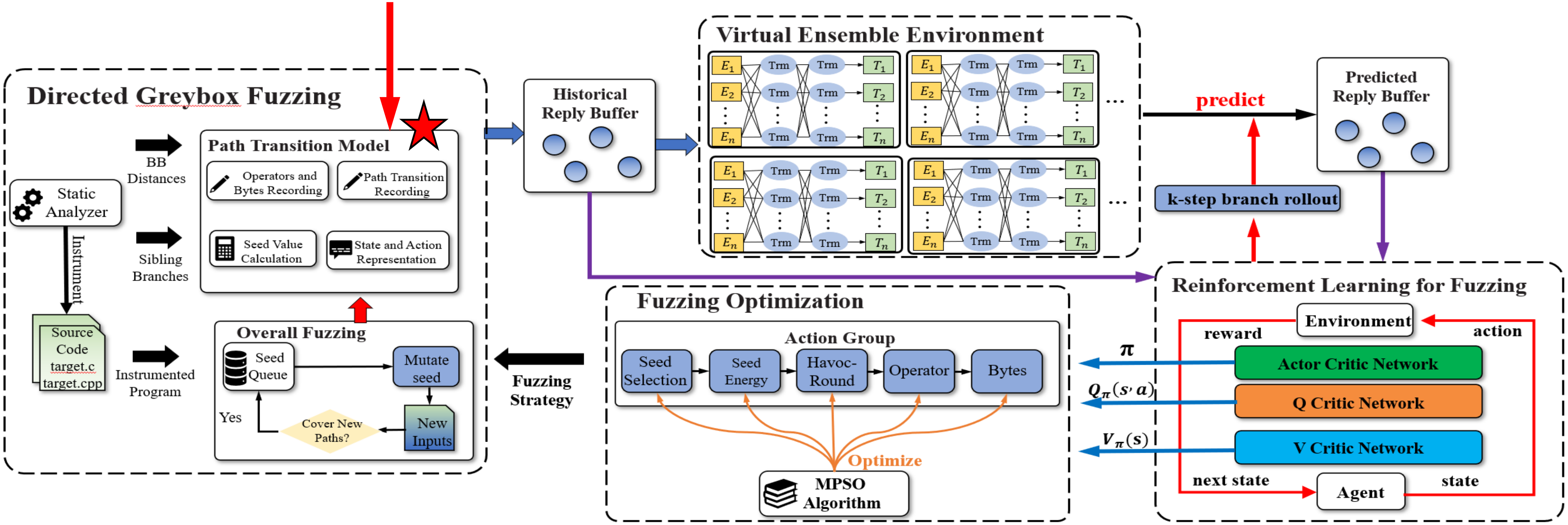
1. Background and Motivation

- **Solutions**
  - ***For Challenge 1***
    - Design the **Virtual Ensemble Environment** to imitate the path transition model and predict the path transitions.
  - ***For Challenge 2***
    - Develop the **Reinforcement Learning for Fuzzing model** to learn the policy that can maximize sequence rewards.
  - ***For Challenge 3***
    - Propose the concept of the **action group** and the **MPSO** algorithm to guide the fuzzer to exercise the optimal path transition sequences

- PART 1 Background and Motivation
- PART 2 Design
- PART 3 Evaluations
- PART 4 Conclusion

2. Design

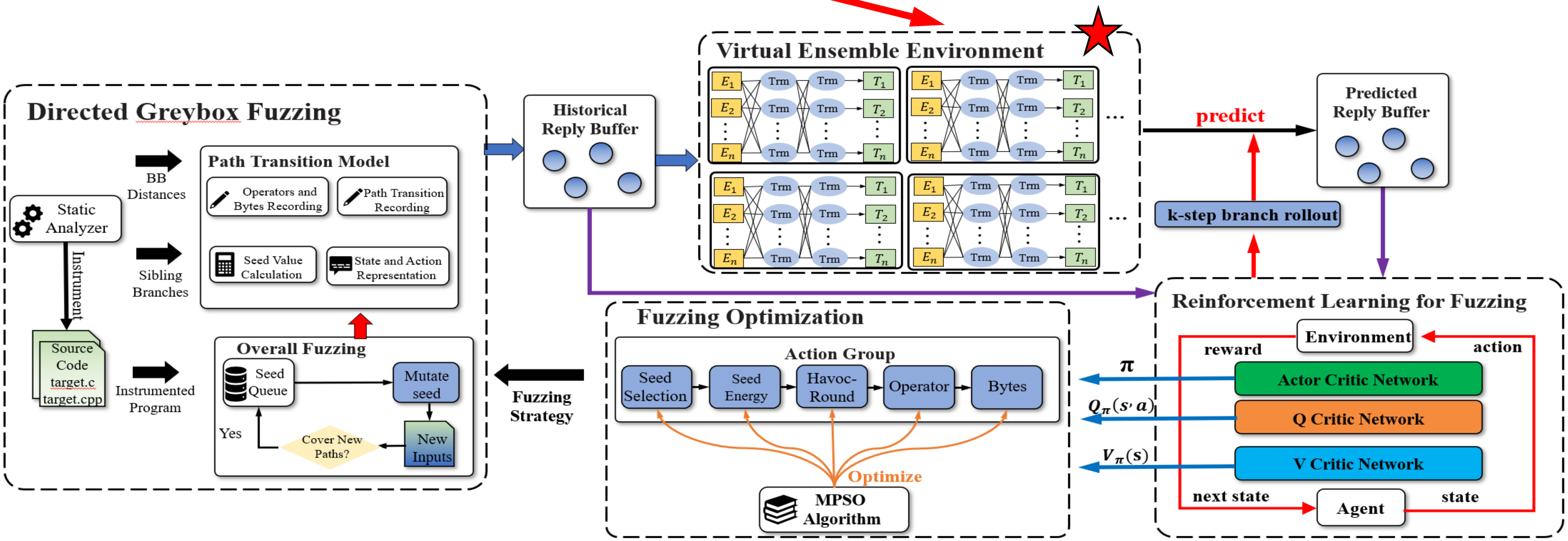
2.1 Overview of DeepGo



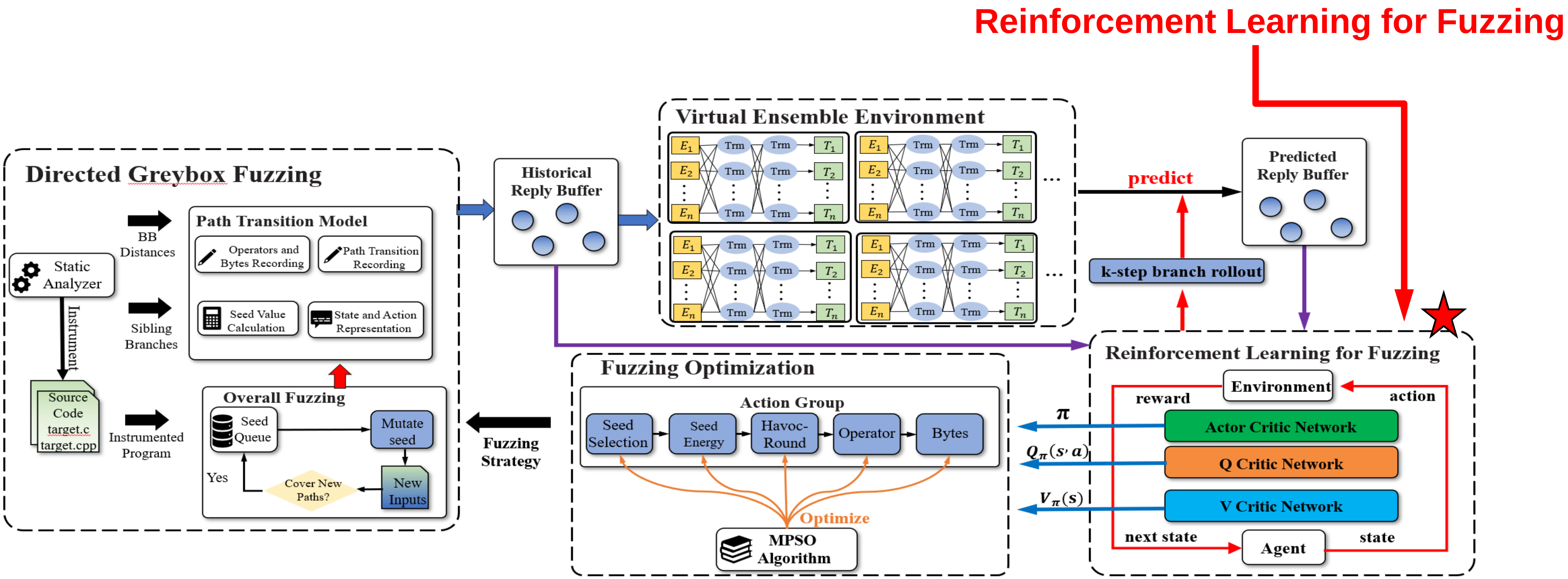
2. Design

2.1 Overview of DeepGo

Virtual Ensemble Environment

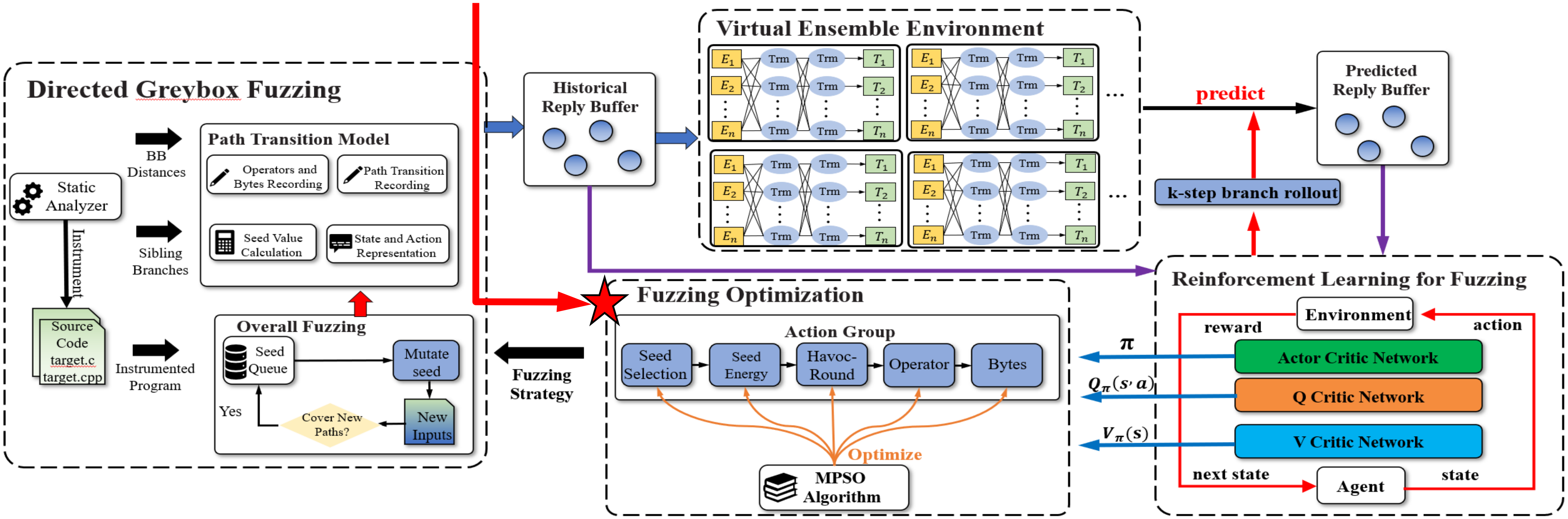


2.1 Overview of DeepGo



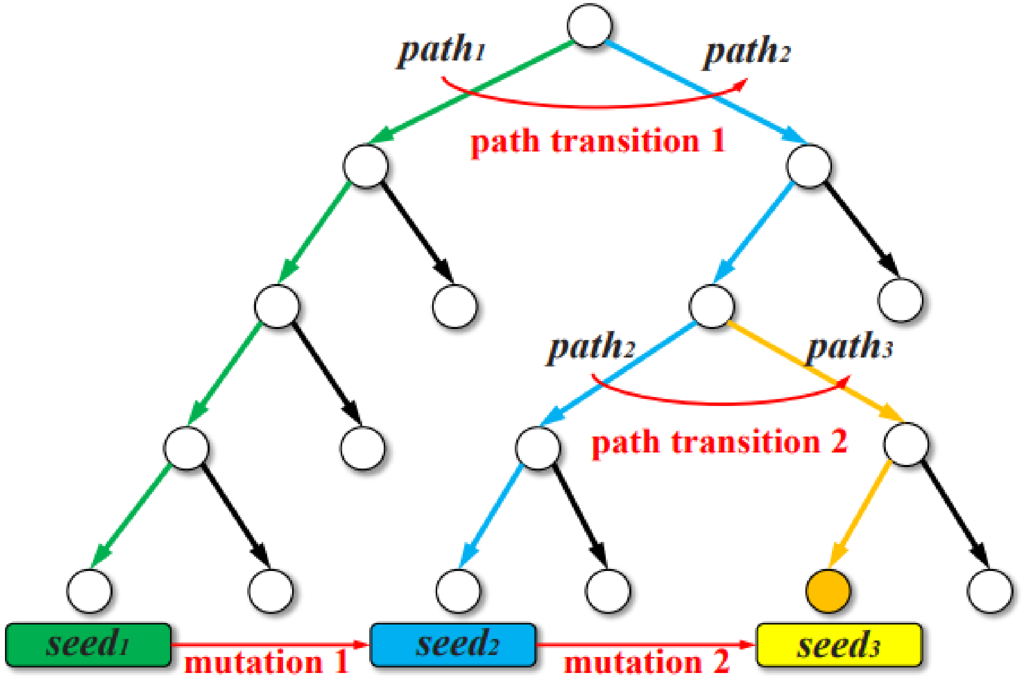
2. Design

2.1 Overview of DeepGo



2. Design

2.2 Design of Path transition model



Reward: effectiveness of path transitions

Expected sequence reward: effectiveness of actions

***Seed value (Path value)*：**

(1) seed distance to targets

(2) the difficulty of satisfying the branch inversion

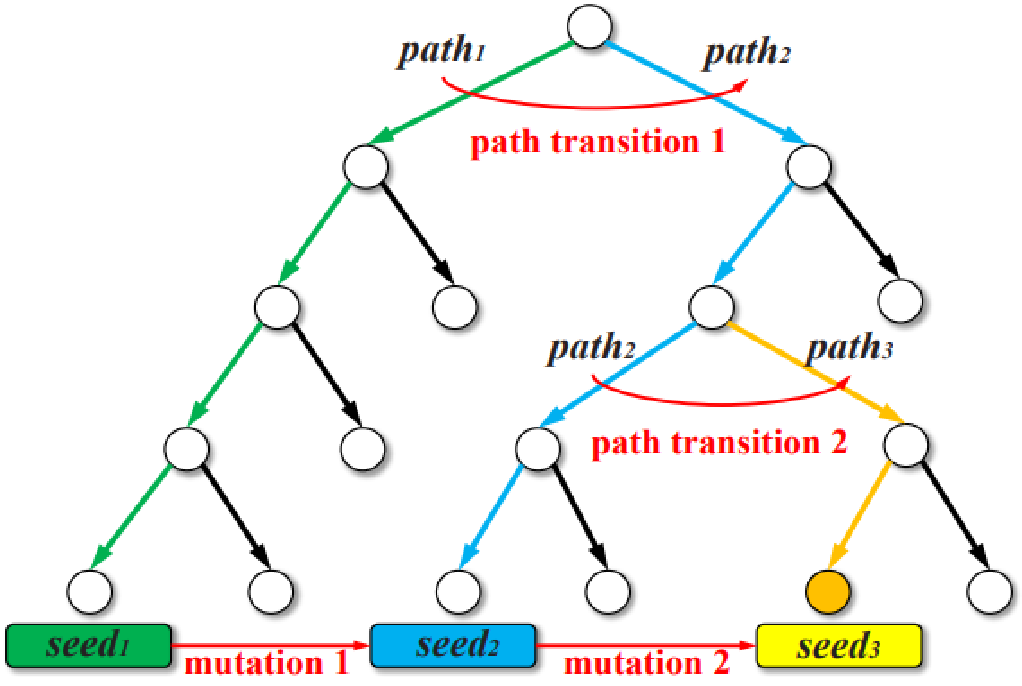
(3) execution speed

(4) “favored”?

$$V^s(p_t) = W_1 \cdot d_s + W_2 \cdot ED_s + W_3 \cdot Ex_s + W_4 \cdot Fv_s$$

2. Design

2.2 Design of Path transition model



Reward:

$$r(p_t, a_t, p_{t+1}) = V^s(p_{t+1}) - V^s(p_t)$$

Path transition:

$$(p_t, a_t, p_{t+1}, r_t)$$

Expected sequence reward:

$$Q_\pi(p, a) = \underset{p' \sim P}{E}[r(p, a, p') + \gamma V_\pi^t(p')]$$

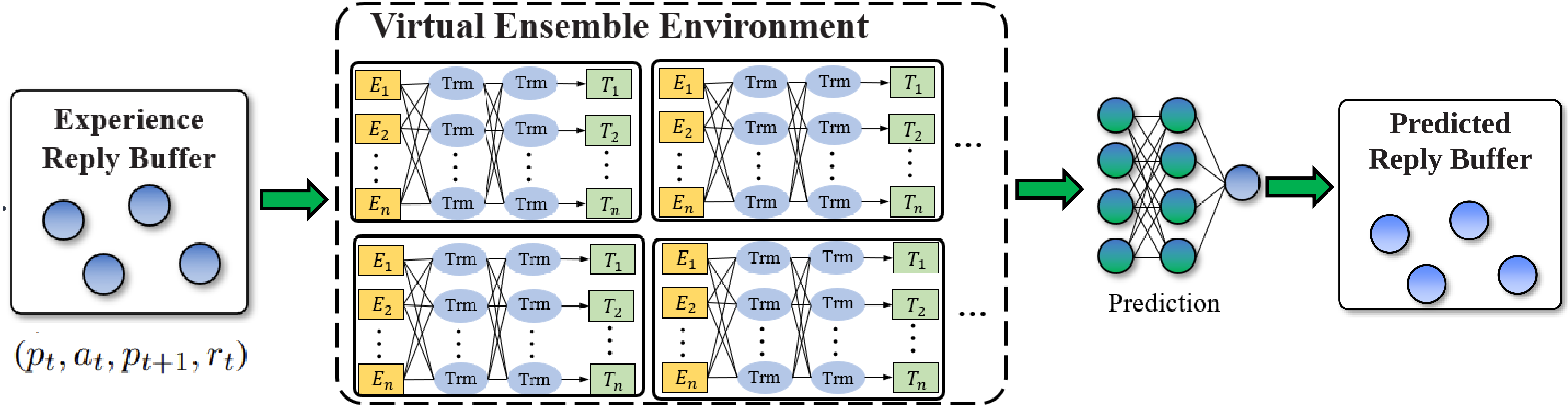
Transition value:

$$V_\pi^t(p') = \begin{cases} 0, & if \;\; p = p_{ter} \\ \sum_a \pi(a|p) \cdot Q_\pi(p', a), & others \end{cases}$$

2. Design

2.3 Design of Virtual Ensemble Environment

- Purpose: predict the potential path transitions and the corresponding rewards.
- Deep neural networks
- Experience reply buffer and predicted replay buffer



2.3 Design of Virtual Ensemble Environment

– Purpose: predict the potential path transitions and the corresponding rewards.

– Deep neural networks

- training: $f{:}(path,\ action) \rightarrow (next\_path,\ reward)$ $\mathbf{X} \longrightarrow \mathbf{Y}$
- Gaussian probability distribution of the next paths and rewards

$$P(p_{t+1}, r_t | p_t, a_t, \theta) = N(\mu_\theta(p_t, a_t), \Sigma_\theta(p_t, a_t))$$

- Average of the probabilities and rewards of DNNs

$$P(p_{t+1}, r_t | p_t, a_t, \theta) = \frac{1}{n} \sum_{i=1}^{n} P(p_{t+1}, r_t | p_t, a_t, \theta_i)$$

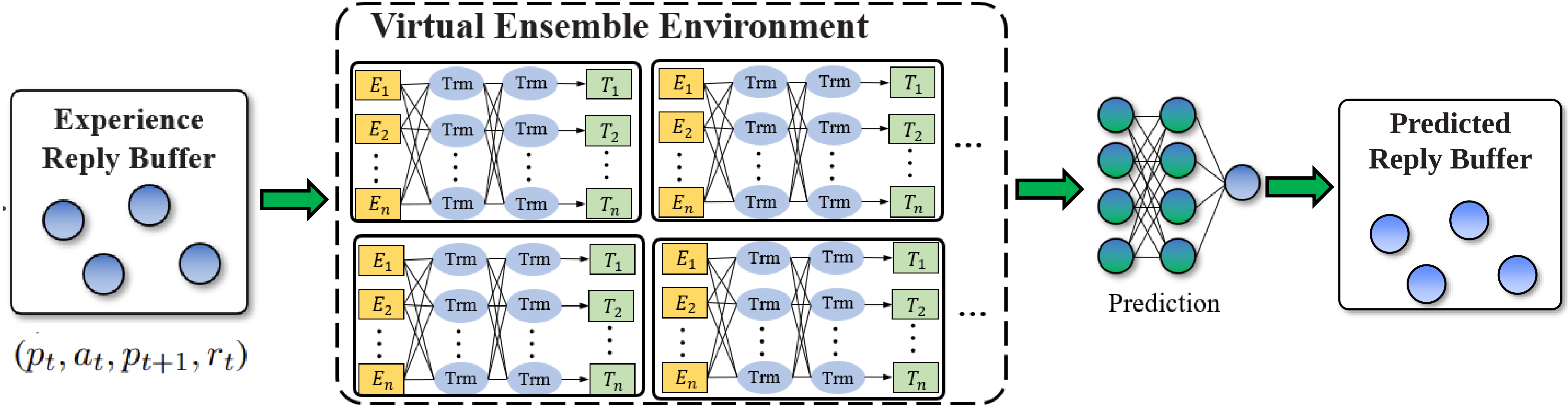
– Experience reply buffer and predicted replay buffer

2.3 Design of Virtual Ensemble Environment

– Purpose: predict the potential path transitions and the corresponding rewards.

– Deep neural networks

– Experience reply buffer and predicted replay buffer

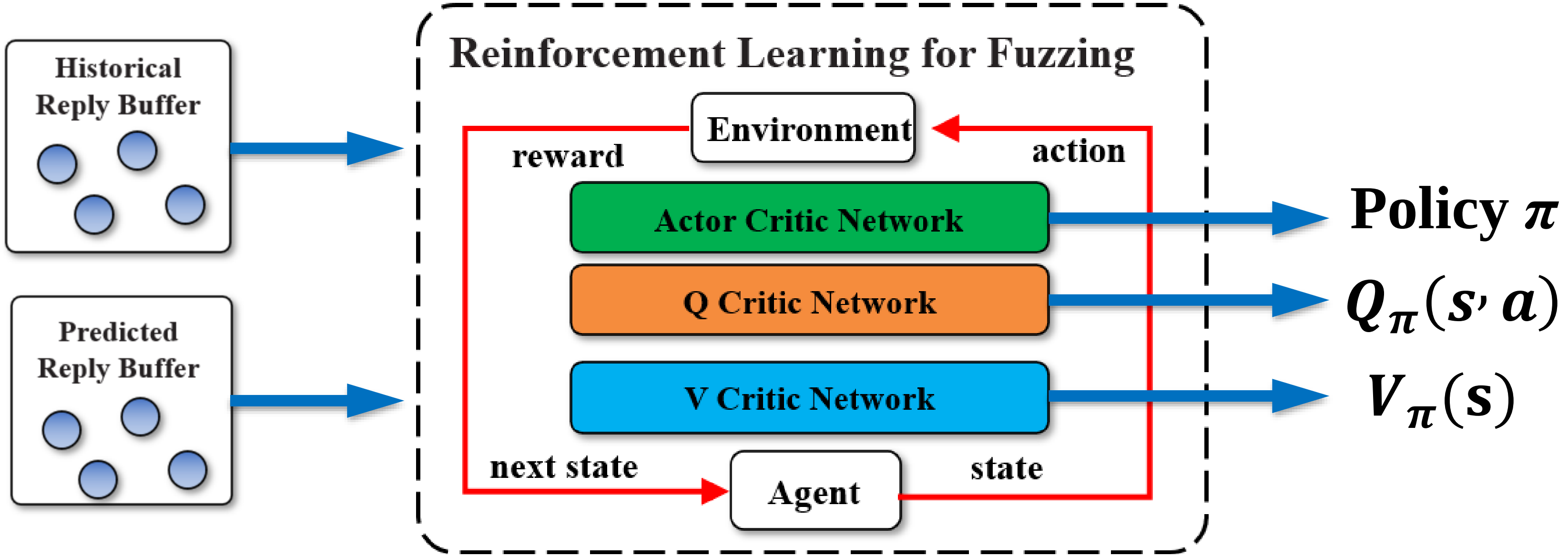


2. Design

2.4 Reinforcement Learning for Fuzzing Model

– Purpose: learn the policy that can steer the fuzzer toward the high-reward path transition sequences

– Actor network, Q-Critic network, V-Critic network

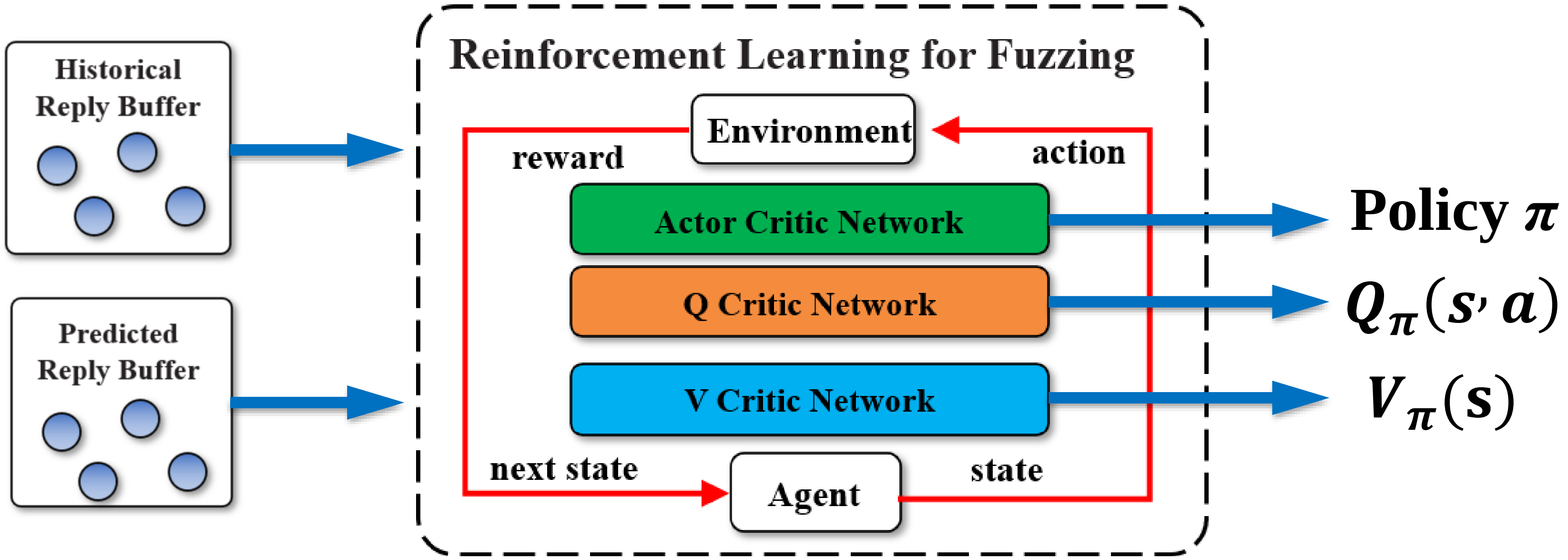


2. Design

2.4 Reinforcement Learning for Fuzzing Model

– Purpose: learn the policy that can steer the fuzzer toward the high-reward path transition sequences

– Actor network, Q-Critic network, V-Critic network



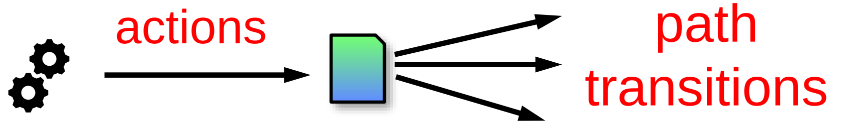
2. Design

2.4 Reinforcement Learning for Fuzzing Model

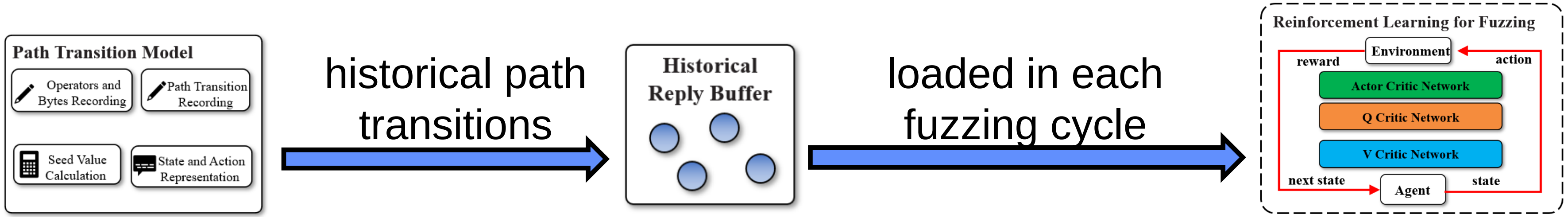
– To give the RLF model foresight, we combine historical path transitions and predicted path transitions to train RLF.

- Historical path transitions:

  - Fuzzer stay on a path and take actions to cause path transitions



  - Historical path transitions are stored in the historical reply buffer and loaded by the RLF model in each fuzzing cycle



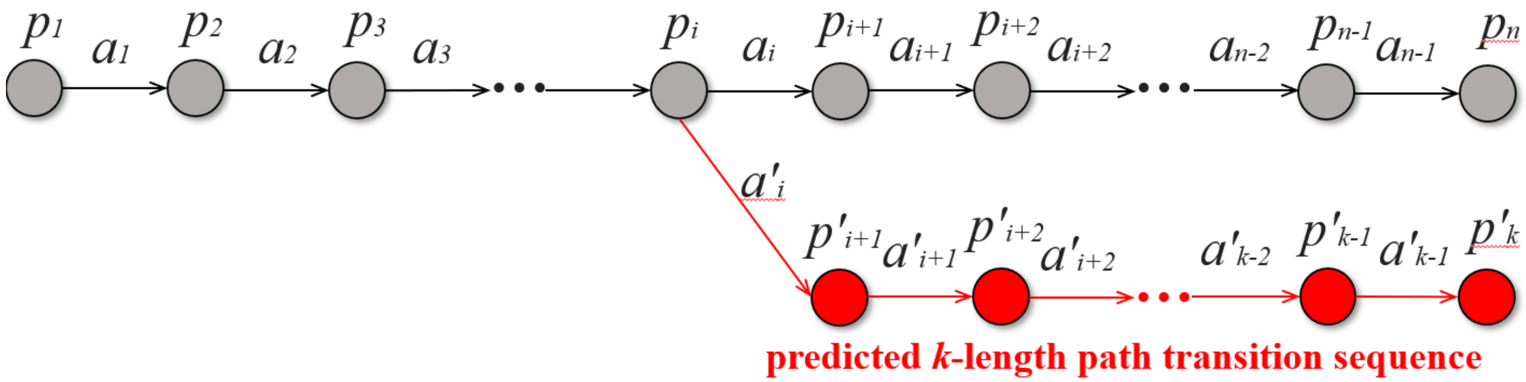
2.4 Reinforcement Learning for Fuzzing Model

– To give the RLF model foresight, we combine historical path transitions and predicted path transitions to train RLF.

- Predicted path transitions:

  - Well-trained VEE imitate path transition model

  - *K*-step branch rollout strategy to obtain predicted path transitions .

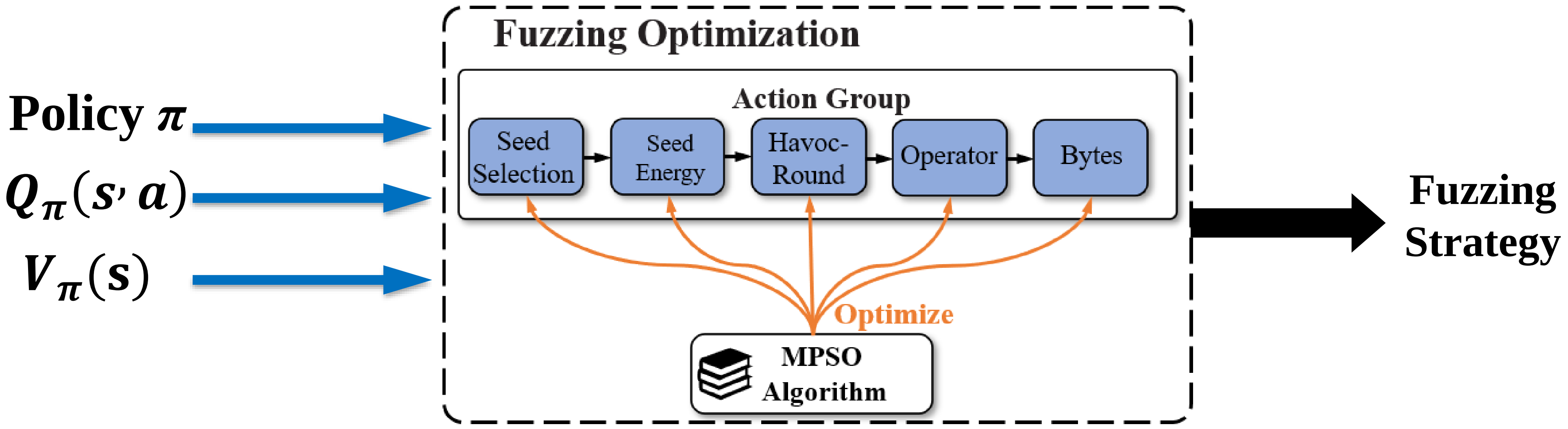


2.5 Fuzzing Optimization

– Purpose: guide the fuzzer to exercise the optimal path transition sequences.

– Action group

– Multi-elements Particle Swarm Optimization (MPSO) algorithm



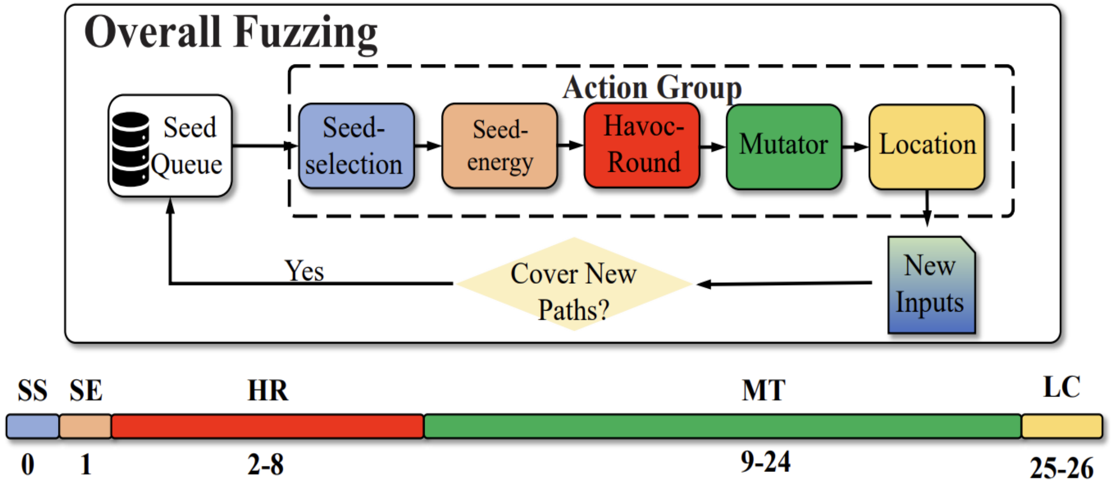
2. Design

2.5 Fuzzing Optimization

– Purpose

– Action group

– MPSO algorithm



Seed-selection (SS):

- Representing the probability of a seed being selected to fuzz.

Seed-energy (SE) :

- Representing the energy assigned to the seed

Havoc-round (HR) :

- Representing the number of looping rounds used to select different mutators and bytes during the havoc stage.

Mutator (MT) :

- Representing the mutator selected to mutate the seed.

Location (LC) :

- Representing the mutation location of the seed that is selected to mutate

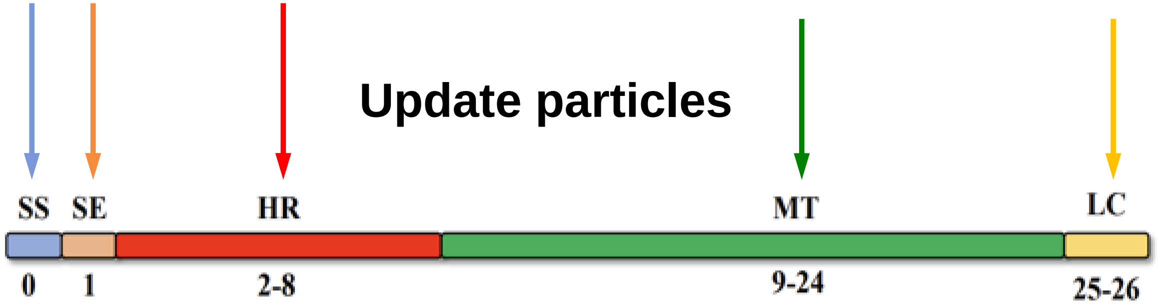
2. Design

2.5 Fuzzing Optimization

– Purpose

– Action group

– Multi-elements Particle Swarm Optimization.



– Update the location to find the local_best and global_best locations for each elements

– Optimize fuzzing strategies to **realize optimal path transition sequences**.

Algorithm 1 MPSO Algorithm

Input: $\Omega_{(s,p)}$
Output: U_s, $\Omega_{(s,p')}$

Initial($\Omega_{(s,p)}$)
while *fuzzing* **do**
  for (s_i, p_i) in $\Omega_{(s,p)}$ **do**
    if $Prob\_Sel_s(p_i(\mathbf{SS}))$ == **True then**
      $mn_i \leftarrow$Cal_MN($p_i(\mathbf{SE})$)
      for j in mn_i **do**
        $hr_j \leftarrow \$Prob\_Sel_h(p_i(\mathbf{HR})$, $hm_j \leftarrow$<>
        for k in hr_j **do**
          $lc_k \leftarrow Prob\_Sel_l(p_i(\mathbf{LC}))$,
          $mt_k \leftarrow Prob\_Sel_m(p_i(\mathbf{MT}))$,
          $hm_j \leftarrow hm_j \cup (lc_k, mt_k)$
        end for
        new_input = Mutate(hm_j, s_i)
        $eff_{local}, eff_{global}$ = Cal_eff(s_i, new_input)
        Update(lbest, gbest, p_i)
      end for
    end if
  end for
end while

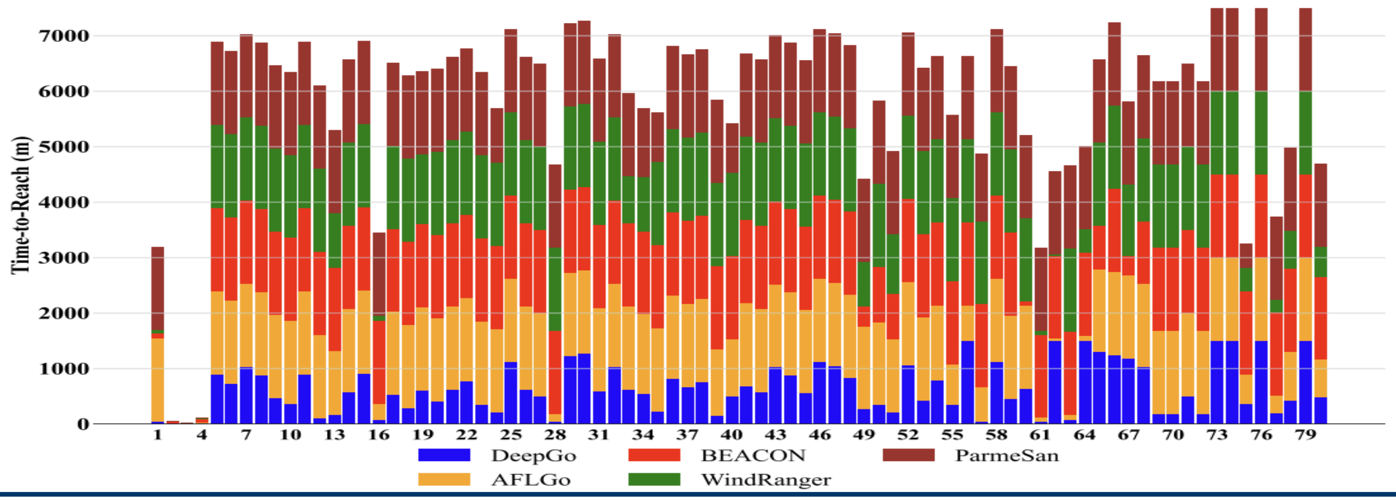
- PART 1 Background and Motivation
- PART 2 Design
- **PART 3 Evaluations**
- PART 4 Conclusion

3. Evaluations

- **Benchmarks**
  - UniBench and CVE-Benchmark
  - **25** programs with a total **100** targets
- **Baselines**
  - WindRanger, BEACON, ParmeSan, and AFLGo
- **Evaluation setup**
  - Repeat **5** times
  - Run for **24** hours

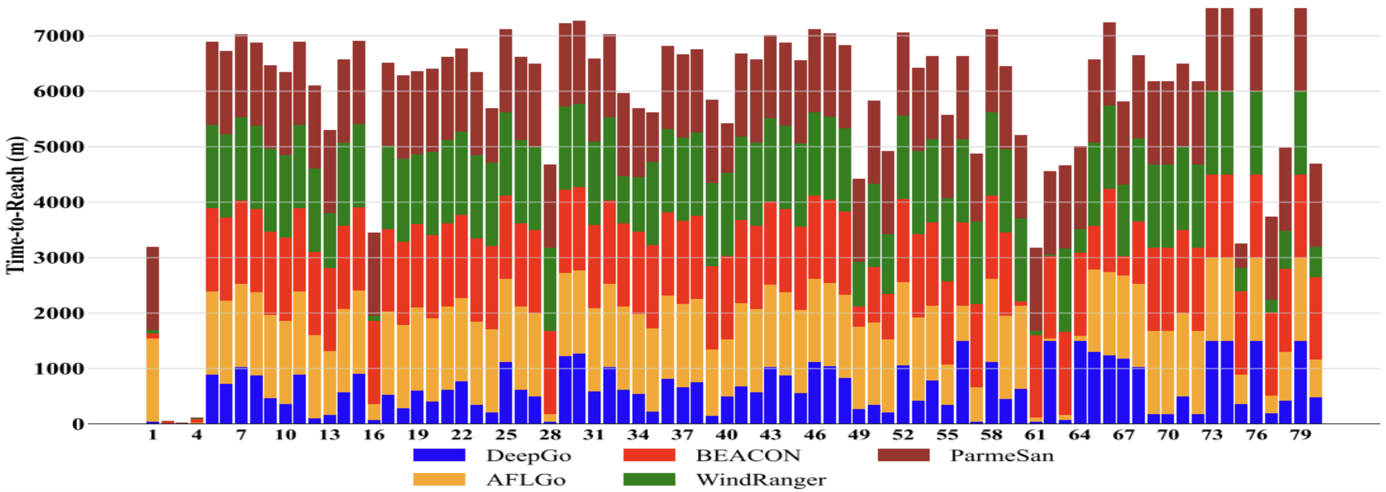
3. Evaluations

- **Time-to-Reach (TTR):**
  - DeepGo can reach the most (**73/80**) target sites compared to AFLGo (**22/80**), BEACON (**11/80**), WindRanger (**19/80**), and ParmeSan (**9/80**) within the time budget.
  - DeepGo demonstrates 3.23×, 1.72×, 1.81×, and 4.83× speedup compared to AFLGo, BEACON, WindRanger, and ParmeSan, respectively



3. Evaluations

- **Time-to-Reach (TTR):**
  - DeepGo can reach the most (73/80) target sites compared to AFLGo (22/80), BEACON (11/80), WindRanger (19/80), and ParmeSan (9/80) within the time budget.
  - DeepGo demonstrates **3.23×**, **1.72×**, **1.81×**, and **4.83×** speedup compared to AFLGo, BEACON, WindRanger, and ParmeSan, respectively.



3. Evaluations

- **Time-to-Exposure(TTE):**
  - DeepGo (**19**) exposed the most compared to AFLGo (**14**), BEACON (**13**), WindRanger (**16**), and ParmeSan (**14**).
  - DeepGo demonstrated 2.61×, 3.32×, 2.43× and 2.53× speedup compared to AFLGo, BEACON, WindRanger, and ParmeSan, respectively.

Prog.	**CVE-ID**	AFLGo	BEACON	WindRa	ParmeS	DeepGo
$\text{binutils}_{2.26}$	2016-4487	2.33m	0.63m	1.21m	0.95m	1.34m
	2016-4488	4.23m	32.1m	3.32m	2.62m	2.69m
	2016-4489	3.36m	2.98m	5.88m	2.31m	1.23m
	2016-4490	1.15m	2.35m	2.63m	0.82m	1.97m
	2016-4491	448m	258m	298m	212m	129m
	2016-4492	10.8m	43.6m	7.47m	4.33m	6.94m
	2016-6131	348m	292m	318m	244m	68.1m
$\text{libming}_{4.48}$	2018-8807	331m	267m	171m	301m	101m
	2018-8962	234m	163m	121m	198m	54.8m
	2018-11095	T.O.	914m	1311m	T.O.	812m
	2018-11225	T.O.	438m	996m	T.O.	128m
$\text{LibPNG}_{1.5.1}$	2011-2501	10.2m	N/A	7.81m	4.53m	3.46m
	2011-3328	69.1m	N/A	49.3m	193m	17.5m
	2015-8540	0.88m	N/A	0.96m	3.41m	5.65m
$\text{xmllint}_{2.9.4}$	2017-9047	T.O.	T.O.	T.O.	T.O.	783m
	2017-9048	T.O.	T.O.	T.O.	T.O.	1389m
	2017-9049	T.O.	T.O.	T.O.	T.O.	T.O.
	2017-9050	T.O.	T.O.	T.O.	T.O.	911m
$\text{Lrzip}_{0.631}$	2017-8846	348m	156m	223m	466m	131m
	2018-11496	201m	98.1m	169m	126m	78.9m
speedup		**2.61×**	**3.32×**	**2.43×**	**2.53×**	-
mean $\hat{A}_{12}$		**0.79**	**0.72**	**0.75**	**0.81**	-
mean p-values		0.018	0.032	0.026	0.011	-

3. Evaluations

- **Time-to-Exposure(TTE):**
  - DeepGo (19) exposed the most compared to AFLGo (14), BEACON (13), WindRanger (16), and ParmeSan (14).
  - DeepGo demonstrated **2.61×, 3.32×, 2.43×** and **2.53×** speedup compared to AFLGo, BEACON, WindRanger, and ParmeSan, respectively.

Prog.	**CVE-ID**	AFLGo	BEACON	WindRa	ParmeS	DeepGo
$\text{binutils}_{2.26}$	2016-4487	2.33m	0.63m	1.21m	0.95m	1.34m
	2016-4488	4.23m	32.1m	3.32m	2.62m	2.69m
	2016-4489	3.36m	2.98m	5.88m	2.31m	1.23m
	2016-4490	1.15m	2.35m	2.63m	0.82m	1.97m
	2016-4491	448m	258m	298m	212m	129m
	2016-4492	10.8m	43.6m	7.47m	4.33m	6.94m
	2016-6131	348m	292m	318m	244m	68.1m
$\text{libming}_{4.48}$	2018-8807	331m	267m	171m	301m	101m
	2018-8962	234m	163m	121m	198m	54.8m
	2018-11095	T.O.	914m	1311m	T.O.	812m
	2018-11225	T.O.	438m	996m	T.O.	128m
$\text{LibPNG}_{1.5.1}$	2011-2501	10.2m	N/A	7.81m	4.53m	3.46m
	2011-3328	69.1m	N/A	49.3m	193m	17.5m
	2015-8540	0.88m	N/A	0.96m	3.41m	5.65m
$\text{xmllint}_{2.9.4}$	2017-9047	T.O.	T.O.	T.O.	T.O.	783m
	2017-9048	T.O.	T.O.	T.O.	T.O.	1389m
	2017-9049	T.O.	T.O.	T.O.	T.O.	T.O.
	2017-9050	T.O.	T.O.	T.O.	T.O.	911m
$\text{Lrzip}_{0.631}$	2017-8846	348m	156m	223m	466m	131m
	2018-11496	201m	98.1m	169m	126m	78.9m
speedup		**2.61×**	**3.32×**	**2.43×**	**2.53×**	-
mean $\hat{A}_{12}$		**0.79**	**0.72**	**0.75**	**0.81**	-
mean p-values		0.018	0.032	0.026	0.011	-

3. Evaluations

- **Ablation study:**
  - Run DeepGo, DeepGo-v and DeepGo-r on UniBench for the TTR experiment
    - DeepGo-v: remove VEE from DeepGo
    - DeepGo-r: remove RLF and FO from DeepGo
  - DeepGo (73/80) can reach much more target sites than DeepGo-v (32/80) and DeepGo-r (18/80), respectively
  - DeepGo outperforms DeepGo-v and DeepGo-r by 2.05× and 3.72×, respectively, in the average TTR of reaching the target sites

3. Evaluations

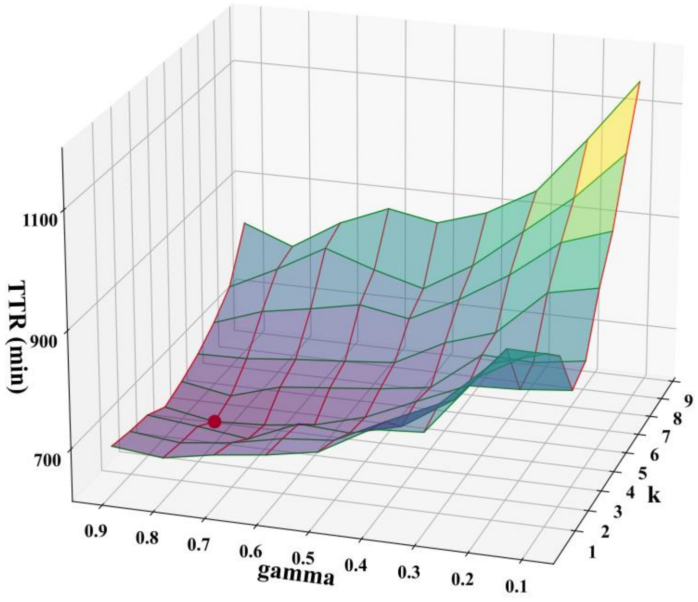
- **Ablation study:**
  - Run DeepGo, DeepGo-v and DeepGo-r on UniBench for the TTR experiment
    - DeepGo-v: remove VEE from DeepGo
    - DeepGo-r: remove RLF and FO from DeepGo
  - DeepGo (**73/80**) can reach much more target sites than DeepGo-v (**32/80**) and DeepGo-r (**18/80**), respectively
  - DeepGo outperforms DeepGo-v and DeepGo-r by 2.05× and 3.72×, respectively, in the average TTR of reaching the target sites

3. Evaluations

- **Ablation study:**
  - Run DeepGo, DeepGo-v and DeepGo-r on UniBench for the TTR experiment
    - DeepGo-v: remove VEE from DeepGo
    - DeepGo-r: remove RLF and FO from DeepGo
  - DeepGo (73/80) can reach much more target sites than DeepGo-v (32/80) and DeepGo-r (18/80), respectively
  - DeepGo outperforms DeepGo-v and DeepGo-r by **2.05×** and **3.72×**, respectively, in the average TTR of reaching the target sites

3. Evaluations

- **Setting of hyperparameters:**
  - Utilize DeepGo with different hyperparameter configurations to test 20 programs from UniBench and recorded the mean TTR for each test case
  - **γ = 0.8** and **k = 4** can achieve minimum TTR
  - The setting of γ and k has a relatively small impact on TTR if the value of **γ is between [0.5, 0.9]**, and the value of **k is between [3, 5]**



- PART 1 Background and Motivation
- PART 2 Design
- PART 3 Evaluations
- **PART 4 Conclusion**

4. Conclusion

- **We propose DeepGo: a predictive directed greybox fuzzer to steer DGF to reach targets via optimal paths**
  - Propose the **path transition model**.
  - Construct a Virtual Ensemble Environment to predict path transitions.
  - Develop a Reinforcement Learning for Fuzzing model to learn the policy that can steer the fuzzer toward the high-reward path transition sequences.
  - Propose the concept of action group and the Multi-elements Particle Swarm Optimization algorithm to steer fuzzer to realize the optimal and viable path transition sequences.

4. Conclusion

- **We propose DeepGo: a predictive directed greybox fuzzer to steer DGF to reach targets via optimal paths**
  - Propose the path transition model.
  - Construct a **Virtual Ensemble Environment** to predict path transitions.
  - Develop a Reinforcement Learning for Fuzzing model to learn the policy that can steer the fuzzer toward the high-reward path transition sequences.
  - Propose the concept of action group and the Multi-elements Particle Swarm Optimization algorithm to steer fuzzer to realize the optimal and viable path transition sequences.

4. Conclusion

- **We propose DeepGo: a predictive directed greybox fuzzer to steer DGF to reach targets via optimal paths**
  - Propose the path transition model.
  - Construct a Virtual Ensemble Environment to predict path transitions.
  - Develop a **Reinforcement Learning for Fuzzing model** to learn the policy that can steer the fuzzer toward the high-reward path transition sequences.
  - Propose the concept of action group and the Multi-elements Particle Swarm Optimization algorithm to steer fuzzer to realize the optimal and viable path transition sequences.

4. Conclusion

- **We propose DeepGo: a predictive directed greybox fuzzer to steer DGF to reach targets via optimal paths**
  - Propose the path transition model.
  - Construct a Virtual Ensemble Environment to predict path transitions.
  - Develop a Reinforcement Learning for Fuzzing model to learn the policy that can steer the fuzzer toward the high-reward path transition sequences.
  - Propose the concept of **action group** and the **Multi-elements Particle Swarm Optimization algorithm** to steer fuzzer to realize the optimal and viable path transition sequences.

Thank you !

If you have some questions about our work,
welcome to contact us!

Email: phlin22@nudt.edu.cn

Artifact of DeepGo: https://gitee.com/paynelin/DeepGo

National University of Defense Technology