core idea

HUỲNH KHÁNH HÒA

September 2025

1 ERL flow

Algorithm 1 Algorithm ERL flow

$D_0 \leftarrow \{0 \mid i \in \{1, \dots, N_{dist}\}\}$
$P_0 \leftarrow \{x_i \sim U(b_{\text{low}}, b_{\text{up}}) \mid i \in \{1, \dots, N_{pop}\}\}$
$P_{0,fitness} \leftarrow \text{Fitness}(P_0)$
$P_{0,i,best} \leftarrow x_i$ ▷ This will clone new
$g_{best} \leftarrow \text{Best}(P_{0,best})$ ▷ This will clone new
while $t \in \{1, \dots, T\}$ **do**
  if $t \mod m == 0$ **then** ▷ This modify directly on x_i
    Run RL with $\text{Best}(P_{t-1,best})$ ▷ it different with g_{best}
  end if
  $D_t \leftarrow \text{Dist}(D_{t-1}, P_{t-1}, g_{best})$
  $P_t \leftarrow \text{Select}(P_{t-1})$
  $P_t \leftarrow \text{Crossover}(P_t)$
  $P_t \leftarrow \text{Mutation}(P_t, D_t)$
  $P_{t,fitness} \leftarrow \text{Fitness}(P_t)$
  $P_{t,i,best} \leftarrow \text{Best}([x_i, P_{t,i,best}])$
  $g_{best} \leftarrow \text{Best}(P_{t,best})$
end while

Algorithm 2 Fitness Evaluation

for $x_i \in P$ **do**
  $F_i \leftarrow f(x_i)$
end for
return F

Algorithm 3 Calculate weight with Rank base

Require: P

$r \leftarrow \{\text{rank}(x_{i,fitness}) \mid \forall x \in P\}$

$z_i \leftarrow \frac{\tau \cdot (r_i - \mu_r)}{\sigma_r}$ ▷ Z-score

$w_i \leftarrow \frac{e^{\beta z_i}}{\sum_{j=1}^{|P|} e^{\beta z_j}}$ ▷ Soft max

return w

Algorithm 4 Selection Operator (Soft Winner Tournament Selection)

$P_{\text{selected}} \leftarrow \emptyset$

while $|P_{\text{selected}}| < |P|$ **do**

  $T \leftarrow \text{Sample}_{\mathcal{U}}(P,\ k)$

  $w \leftarrow \text{RankWeight}(T)$

  $x_{\text{soft}} \leftarrow \sum_{i=1}^{k} w_i \cdot x_i$

  $P_{\text{selected}} \leftarrow P_{\text{selected}} \cup x_{\text{soft}}$

end while

return P_{selected}

Algorithm 5 Distribution (Particle Swarm Optimization)

$D \leftarrow \{0, D_i, \ldots, D_{N-1} \mid i \in \{1, \ldots, N_{dist}\}\}$

$w_{per} \leftarrow \text{RankWeight}(P)$

$w_{cog} \leftarrow \text{RankWeight}(P_{best})$

$D_0 \leftarrow \sum_{i=1}^{|P|} \phi_{per} \cdot w_{per,i} \cdot x_i + \phi_{cog} \cdot w_{cog,i} \cdot x_{i,best} + \phi_{soc} \cdot g_{best}$

return D

Algorithm 6 Estimate Eta Factor

Require: $x \in \mathbb{R}^d$

if $d == 1$ **or** $\max(x) \geq 1$ **then**

  $x' \leftarrow \mathbf{1}$

else

  $x' \leftarrow \log_{10}(\max(|x|))$

  $x'_i \leftarrow \begin{cases} \mathbf{0} & \text{if } x'_i \to \infty \\ x'_i & \text{otherwise} \end{cases}$

  $x' \leftarrow \lceil x' \rceil$

  $x' \leftarrow \frac{x'}{10^{x'}}$

end if

return x'

Algorithm 7 Crossover Operator (Simulated Binary Crossover with Bounds)

$P_{\text{offspring}} \leftarrow \emptyset$

while $|P_{\text{offspring}}| < |P|$ **do**

$\quad x_1, x_2 \leftarrow Sample_{\mathcal{U}}(P, 2)$

$$x_{\min} \leftarrow \min(x_1,\ x_2), x_{\max} \leftarrow \max(x_1,\ x_2)$$

$\quad u \sim \mathcal{U}(0, 1)$

$\quad \eta_c \leftarrow \eta_{cx} \cdot \text{EstimateEtaFractor}(JSD(x_1, x_2))$

$\quad$ **for** $b \in \{L,\ U\}$ **do**

$\quad\quad \alpha_b \leftarrow \begin{cases} 2 - (1 + 2 \cdot \frac{x_{\min} - L}{x_{\max} - x_{\min}})^{-(\eta_c + 1)} & \text{if } b = L \\ 2 - (1 + 2 \cdot \frac{U - x_{\max}}{x_{\max} - x_{\min}})^{-(\eta_c + 1)} & \text{if } b = U \end{cases}$

$\quad\quad \beta_{q,b} \leftarrow \begin{cases} (u \cdot \alpha_b)^{\frac{1}{\eta_c + 1}} & \text{if } u \leq \frac{1}{\alpha_b} \\ \left(\frac{1}{2 - u \cdot \alpha_b}\right)^{\frac{1}{\eta_c + 1}} & \text{otherwise} \end{cases}$

$\quad\quad c_b \leftarrow \begin{cases} 0.5\left[(1 + \beta_{q,b}) \cdot x_1 + (1 - \beta_{q,b}) \cdot x_2\right] & \text{if } b = L \\ 0.5\left[(1 - \beta_{q,b}) \cdot x_1 + (1 + \beta_{q,b}) \cdot x_2\right] & \text{if } b = U \end{cases}$

$\quad\quad c_b \leftarrow \min\left(\max(c_b,\ L),\ U\right)$

$\quad$ **end for**

$\quad P_{\text{offspring}} \leftarrow P_{\text{offspring}} \cup \begin{cases} \{c_U,\ c_L\} & \text{if } \mathcal{U}(0, 1) \leq 0.5 \\ \{c_L,\ c_U\} & \text{otherwise} \end{cases}$

end while

return $P_{\text{offspring}}$

Algorithm 8 Mutation Operator (Polynomial Mutation with Bounds)

1: $P_{\text{offspring}} \leftarrow \emptyset$
2: $w_d \leftarrow \text{RankWeight}(|D|, \ldots, 1)$ ▷ RankWeight with decrease at line 1
3: $d \leftarrow \sum_{i=1}^{|D|} w_{d,i} \cdot D_i$
4: $jsd_P \leftarrow JSD(P)$
5: $jsd_D \leftarrow JSD(D)$ ▷ w apply in JSD for calc M similar with w_d
6: $jsd_{DP} \leftarrow \mathcal{U}(0,1) \cdot mean(jsd_D, jsd_P)$
7: **for** $x_i \in P$ **do**
8:

$$\delta_1 \leftarrow \frac{x_i - L}{U - L}, \delta_2 \leftarrow \frac{U - x_i}{U - L}$$

9: $u \sim \mathcal{U}(0,1)$
10: $\eta_m \leftarrow \eta_{mut} \cdot \text{EstimateEtaFractor}(mean(JSD(x_i, d), jsd_D, jsd_P))$
11:

$$\delta_q \leftarrow \begin{cases} \left[2u + (1-2u) \cdot (1-\delta_1)^{\eta_m+1}\right]^{\frac{1}{\eta_m+1}} - 1 & \text{if } u \le 0.5 \\ 1 - \left[2(1-u) + 2(u-0.5) \cdot (1-\delta_2)^{\eta_m+1}\right]^{\frac{1}{\eta_m+1}} & \text{otherwise} \end{cases}$$

12: $x_i' \leftarrow x_i + \delta_q \cdot (U - L)$
13: $x_i' \leftarrow \min(\max(x_i',\ L),\ U)$
14: $x_i' \leftarrow (1 - jsd_{DP}) \cdot x_i' + jsd_{DP} \cdot d$
15: $P_{\text{offspring}} \leftarrow P_{\text{offspring}} \cup x_i'$
16: **end for**
17: **return** $P_{\text{offspring}}$

Note:

- in compute D: $\phi_{per} + \phi_{cog} + \phi_{soc} = 1$, similar with velocity in PSO: $v_{i,d} \leftarrow wv_{i,d} + \phi_p r_p(p_{i,d} - x_{i,d}) + \phi_g r_g(g_d - x_{i,d})$ but using $v_{i,d}$ without recursive, init with 0, and same v for all (computed with the contribute follow the RankWeight), d in line 3 of mut op can understand as v with sliding window not over the time

- in compute JSD(X): x^d will be scaled into [0,1] & sum(X) = 1; w if don't have any comment mean using uniform distribution as default

- $f(x)$ is the rewards in this paper `https://openreview.net/forum?id=N0I2RtD8je`. But different with the paper is not only use vec in env for training in SAC but also use the img too & using SigLIP2 instead CLIP.

- The NN using from this paper `https://arxiv.org/abs/1703.01513`

- Both arch & model params use same in EA flow but with some different:

+ model params x have search space
$[min(x) + \mathcal{U}(-0.001, 0.001), max(x) + \mathcal{U}(-0.001, 0.001)]$

+ arch have search space [0,1] and in the end of operators use output as probation $\sim \mathcal{B}er(x)$ to get back {0,1}