i) The entire algorithm:

Input: Population size N, $\{\pi_1, \pi_2 .. \pi_N\}$ are the target policies to be evaluated, each represented by a neural network.

For g = 1, 2, .. G generations do:
Initialise an array B to all 0's;
Choose a behaviour policy π using Algorithm 3;
~~Cut~~
For i = 1, 2, N do:
Calculate $L_\pi(\pi_i) \leftarrow E\left[\frac{\pi_i(a|s)}{\pi(a|s)} \cdot Q_\pi(a,s)\right]$;

$S_\pi(\pi_i) \leftarrow L_\pi(\pi_i) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{max}(\pi, \pi_i)$;

// where $\epsilon = \max_{s,a} |A_\pi(s,a)|$, γ = discount-factor

$B[i] \leftarrow S_\pi(\pi_i)$;

Sort B in descending order;
Select the first k policies as parents for next generation
Let P_j^g be the weight matrix of the j^{th} parent in the g^{th} generation
For j = 1, 2 .. N do:
$P_j^{g+1} = P_j^g + \sigma\epsilon$ // $\sigma \rightarrow$ mutation factor
(Generate n-1 children by selecting a random parent every time, ~~and~~ from the top k parents, and tweak α every time)

To be replaced by Crossover()

Add an elite parent at the end
(Elite parent: ~~Steps:~~ The best performing policy, after 'l' runs)
~~i) Select m policies among~~