The optimization problem can be formulated as

$$\min_{\substack{\lambda_0 \\ \lambda_k \geq 0, 1 \leq k \leq K}} \left[\max_{\pi} \mathbf{E} \sum_{t=0}^{T-1} \gamma^t \hat{\mathcal{R}}_{\lambda_0,\ldots,\lambda_K}(s_t, \tilde{a}_t) + \sum_{k=1}^{K} \lambda_k \gamma^T \delta_k \right], \tag{0.1}$$

where the penalized reward function

$$\hat{\mathcal{R}}_{\lambda_0,\ldots,\lambda_K}(s_t, \tilde{a}_t) = \mathcal{R}(s_t, a_t) - \lambda_0 \sum_{j=1}^{J} \mathcal{H}_j(s_t) - \sum_{k=1}^{K} \lambda_k \mathbf{1}(\mathcal{C}_{k,t}(\tilde{a}_t) = 1). \tag{0.2}$$

The parameters we need to choose before running:

(i) *hard-constraints* (step-wise) : h_j, $1 \leq j \leq J$

(ii) *soft-constraint* (episode-wise) : c_k, $1 \leq k \leq K$

(iii) *soft-constraint-tolerance* (optional): t_k, $1 \leq k \leq K$

(iv) *soft-violation-risk-threshold* : r_k, $1 \leq k \leq K$

(v) *hard-lambda-learning-rate* : α

(vi) *soft-lambda-learning-rate* : β_k, $1 \leq k \leq K$

(vii) *number of policy exploitation* (outer loop) : n_p

(viii) *number of updates on lambda* (inner loop) n_λ

(ix) *exploitation-steps* : s_e

(x) *lambda-update-steps* : s_λ

(xi) *last-training-steps* : s_l

(xii) *lambda-sample-size* : m_λ

Algorithm 1: PPO algorithm on updating policy with fixed $\lambda = (\lambda_0, ..., \lambda_K)$

Input: h_j, $1 \leq j \leq J$; c_k, t_k, r_k, $1 \leq k \leq K$; number of steps s; policy parameter θ; value function parameter ϕ.

for $k = 0, 1, ..., s-1$ **do**

Generate a random environment (task) parameter σ.

Collect set of episode(s) D by sampling using policy π_θ in environment $\mathcal{E}_\sigma$.

Compute penalized rewards-to-go $\tilde{\mathcal{R}}_t$ w.r.t. λ_k, $0 \leq k \leq K$.

Compute advantage estimates A_t based on the current value function V_ϕ.

Update the policy parameter by maximizing the PPO-Clip objective:

$$\theta \leftarrow \arg\max_{\theta'} \frac{1}{|D|T} \sum_{\tau \in D} \sum_{t=0}^{T-1} \min \left(\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} A_t, g(\varepsilon, A_t) \right), \tag{0.3}$$

using MLP.

Update the value function parameter ϕ by minimizing the mean square error:

$$\phi \leftarrow \arg\min_{\phi'} \frac{1}{|D|T} \sum_{\tau \in D} \sum_{t=0}^{T-1} \min \left(V_{\phi'}(s_t) - \tilde{\mathcal{R}}_t \right)^2, \tag{0.4}$$

using MLP.

end for

Algorithm 2: Algorithm on updating on $\lambda = (\lambda_0, ..., \lambda_K)$

Input: Numbers of steps, sample size, and constraints.
Initialize $\lambda = 0$, policy parameter θ, and value function parameter ϕ.
for $i = 0, 1, ..., n_p - 1$ **do**
    Apply Algorithm 1 with θ, ϕ, λ, and $s = s_e$.
    for $n = 0, 1, ..., n_\lambda - 1$ **do**
        Apply Algorithm 1 with θ, ϕ, λ, and $s = s_\lambda$.
        for $m = 0, 1, ..., m_\lambda - 1$ **do**
            Generate a random environment (task) parameter σ.
            Collect an episode by sampling using policy π_θ in environment $\mathcal{E}_\sigma$.
            Compute the number of hard violations H^m and the indicator C_k^m of soft violations, $1 \leq k \leq K$.
        end for
        Compute the gradient w.r.t. λ_k, $0 \leq k \leq K$:

$$\nabla_\lambda \leftarrow -\frac{1}{m_\lambda} \sum_{m=0}^{m_\lambda - 1} (H^m, C_1^m - \delta_1, ..., C_K^m - \delta_K). \tag{0.5}$$

        $\lambda \leftarrow \lambda - \mathrm{diag}(\alpha, \beta_1, ..., \beta_K) \nabla_\lambda$.
    end for
end for
Apply Algorithm 1 with θ, ϕ, λ, and $s = s_l$.