PPO for Combinatorial Optimization: A Complete Mathematical Tutorial

---

Table of Contents

---

1. Introduction to PPO

Proximal Policy Optimization (PPO) is a policy gradient method that maintains a balance between exploration and exploitation while ensuring stable updates. For combinatorial optimization, we formulate the problem as a Markov Decision Process (MDP) where an agent constructs a solution step by step.

Why PPO for Combinatorial Optimization?

- **Sequential decision making**: Build solutions incrementally
- **Learns heuristics**: Generalizes patterns across instances
- **No ground truth needed**: Only requires reward signal
- **Handles constraints**: Through reward shaping or masking

---

2. PPO Mathematical Foundation

2.1 Markov Decision Process (MDP)

We define an MDP as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action space

- $\mathcal{P}(s_{t+1}|s_t, a_t)$: Transition probability (deterministic for our problems)
- $\mathcal{R}(s_t, a_t)$: Reward function
- $\gamma \in [0, 1]$: Discount factor

2.2 Policy and Value Functions

Policy $\pi_\theta(a|s)$: Probability of taking action a in state s, parameterized by θ.

State Value Function:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \,\middle|\, s_0 = s \right]$$

Action Value Function:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a \right]$$

Advantage Function:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

The advantage tells us how much better action a is compared to the average action under policy π.

2.3 Policy Gradient Theorem

The gradient of the expected return with respect to policy parameters:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^{\pi_\theta}(s_t, a_t) \right]$$

2.4 Generalized Advantage Estimation (GAE)

To reduce variance while maintaining acceptable bias, we use GAE:

TD Residual:

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

GAE:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l}$$

Expanded form:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots + (\gamma\lambda)^{T-t}\delta_T$$

Where:

- $\lambda = 0$: One-step TD (high bias, low variance)
- $\lambda = 1$: Monte Carlo (low bias, high variance)
- $\lambda \in (0, 1)$: Balanced trade-off (typically $\lambda = 0.95$)

2.5 PPO Clipped Objective

Probability Ratio:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Clipped Surrogate Objective:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

Where ϵ is the clipping parameter (typically $\epsilon = 0.2$).

Intuition:

- If $\hat{A}_t > 0$ (good action): We want to increase $\pi_\theta(a_t|s_t)$, but clip prevents $\rho_t > 1 + \epsilon$
- If $\hat{A}_t < 0$ (bad action): We want to decrease $\pi_\theta(a_t|s_t)$, but clip prevents $\rho_t < 1 - \epsilon$

2.6 Value Function Loss

$$\mathcal{L}^{\text{VF}}(\phi) = \mathbb{E}_t \left[\left(V_\phi(s_t) - \hat{R}_t \right)^2 \right]$$

Where $\hat{R}_t$ is the target return:

$$\hat{R}_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$$

Or with bootstrapping:

$$\hat{R}_t = \hat{A}_t + V_{\phi_{\text{old}}}(s_t)$$

2.7 Entropy Bonus

To encourage exploration:

$$\mathcal{H}[\pi_\theta(\cdot|s_t)] = -\sum_{a \in \mathcal{A}(s_t)} \pi_\theta(a|s_t) \log \pi_\theta(a|s_t)$$

$$\mathcal{L}^{\text{ENT}}(\theta) = -\mathbb{E}_t \left[\mathcal{H}[\pi_\theta(\cdot|s_t)] \right]$$

2.8 Total PPO Loss

$$\mathcal{L}^{\text{PPO}}(\theta, \phi) = -\mathcal{L}^{\text{CLIP}}(\theta) + c_1 \mathcal{L}^{\text{VF}}(\phi) - c_2 \mathcal{H}[\pi_\theta]$$

Typical values: $c_1 = 0.5, c_2 = 0.01$

---

3. Traveling Salesman Problem (TSP)

3.1 Problem Definition

Given N cities with coordinates $\mathbf{P} = \{p_1, p_2, \ldots, p_N\}$ where $p_i \in \mathbb{R}^2$, find a tour $\tau = (c_0, c_1, \ldots, c_{N-1}, c_0)$ that visits each city exactly once and minimizes total distance:

$$L(\tau) = \sum_{t=0}^{N-1} d(p_{c_t}, p_{c_{t+1}}) + d(p_{c_{N-1}}, p_{c_0})$$

where $d(p_i, p_j) = \|p_i - p_j\|_2$ is Euclidean distance.

3.2 MDP Formulation

State Space

At time step t, the state is:

$$s_t = (\mathbf{P}, \mathbf{v}_t, c_t, c_0)$$

Where:

- $\mathbf{P} \in \mathbb{R}^{N \times 2}$: City coordinates (static)
- $\mathbf{v}_t \in \{0, 1\}^N$: Visited mask, $v_t^{(i)} = 1$ if city i visited
- $c_t \in \{1, \ldots, N\}$: Current city index
- c_0: Starting city (for return trip calculation)

Action Space

$$\mathcal{A}(s_t) = \{i \in \{1, \ldots, N\} : v_t^{(i)} = 0\}$$

The agent selects the next unvisited city.

Transition Dynamics

Deterministic transitions:

$$s_{t+1} = \mathcal{T}(s_t, a_t)$$

$$\mathbf{v}_{t+1}^{(i)} = \begin{cases} 1 & \text{if } i = a_t \text{ or } v_t^{(i)} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$c_{t+1} = a_t$$

3.3 Policy Network Architecture

Node Embedding

Initial embedding for each city:

$$\mathbf{h}_i^{(0)} = \text{Linear}(p_i) \in \mathbb{R}^d$$

Transformer Encoder (L layers)

For layer $l = 0, 1, \ldots, L-1$:

Multi-Head Self-Attention:

$$\mathbf{Q}^{(l)} = \mathbf{H}^{(l)}\mathbf{W}_Q, \quad \mathbf{K}^{(l)} = \mathbf{H}^{(l)}\mathbf{W}_K, \quad \mathbf{V}^{(l)} = \mathbf{H}^{(l)}\mathbf{W}_V$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

Feed-Forward:

$$\mathbf{H}^{(l+1)} = \text{LayerNorm}\left(\mathbf{H}^{(l)} + \text{FFN}\left(\text{LayerNorm}\left(\mathbf{H}^{(l)} + \text{MHA}(\mathbf{H}^{(l)})\right)\right)\right)$$

Context Vector

Combine global and local information:

$$\mathbf{h}_{\text{context}} = \left[\bar{\mathbf{h}} \,\|\, \mathbf{h}_{c_t} \,\|\, \mathbf{h}_{c_0}\right]$$

Where $\bar{\mathbf{h}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{h}_i^{(L)}$ is the graph embedding.

Action Probabilities

Query from context:

$$\mathbf{q} = \text{Linear}(\mathbf{h}_{\text{context}})$$

Keys from unvisited cities:

$$\mathbf{k}_i = \text{Linear}(\mathbf{h}_i^{(L)})$$

Logits with masking:

$$u_i = \begin{cases} \frac{\mathbf{q}^\top \mathbf{k}_i}{\sqrt{d}} & \text{if } v_t^{(i)} = 0 \\ -\infty & \text{if } v_t^{(i)} = 1 \end{cases}$$

Policy:

$$\pi_\theta(a_t = i \mid s_t) = \frac{\exp(u_i)}{\sum_{j:v_t^{(j)}=0} \exp(u_j)}$$

3.4 Reward Function

Option A: Sparse Terminal Reward

$$r_t = \begin{cases} -L(\tau) & \text{if } t = N-1 \text{ (terminal)} \\ 0 & \text{otherwise} \end{cases}$$

Option B: Dense Step Reward

$$r_t = -d(p_{c_t}, p_{a_t})$$

At terminal step, add return distance:

$$r_{N-1} = -d(p_{c_{N-1}}, p_{c_0})$$

Option C: Normalized Dense Reward (Recommended)

$$r_t = -\frac{d(p_{c_t}, p_{a_t})}{D_{\text{avg}}}$$

Where:

$$D_{\text{avg}} = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} d(p_i, p_j)$$

This normalization helps when training across instances of varying sizes.

3.5 Value Network

$$V_\phi(s_t) = \text{MLP}\left(\left[\bar{\mathbf{h}} \,\|\, \mathbf{h}_{c_t} \,\|\, \frac{t}{N} \,\|\, \frac{L_{\text{partial}}}{D_{\text{avg}} \cdot N} \right] \right)$$

Where $L_{\text{partial}} = \sum_{k=0}^{t-1} d(p_{c_k}, p_{c_{k+1}})$ is the partial tour length.

3.6 Complete PPO for TSP

Collect trajectories: Run policy to get $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^{N-1}$

Compute returns:

$$\hat{R}_t = \sum_{l=0}^{N-1-t} \gamma^l r_{t+l}$$

Compute advantages (GAE):

$$\hat{A}_t = \sum_{l=0}^{N-1-t} (\gamma\lambda)^l \left(r_{t+l} + \gamma V_\phi(s_{t+l+1}) - V_\phi(s_{t+l})\right)$$

Update (for K epochs):

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}^{\text{CLIP}}(\theta)$$

$$\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}^{\text{VF}}(\phi)$$

---

4. Knapsack Problem

4.1 Problem Definition

Given N items, each with value $v_i > 0$ and weight $w_i > 0$, and a knapsack with capacity $C_{\max}$, select a subset $S \subseteq \{1, \ldots, N\}$ to maximize total value:

$$\max_S \sum_{i \in S} v_i \quad \text{subject to} \quad \sum_{i \in S} w_i \leq C_{\max}$$

4.2 MDP Formulation

State Space

$$s_t = (\mathbf{x}_t, W_t, V_t, t)$$

Where:

- $\mathbf{x}_t \in \{-1, 0, 1\}^N$: Item status
  - $x_t^{(i)} = 1$: Item i selected
  - $x_t^{(i)} = 0$: Item i undecided
  - $x_t^{(i)} = -1$: Item i rejected

- $W_t = \sum_{i:x_t^{(i)}=1} w_i$: Current total weight
- $V_t = \sum_{i:x_t^{(i)}=1} v_i$: Current total value
- t: Current step (which item to decide)

Action Space

For sequential decision (one item per step):

$$\mathcal{A}(s_t) = \{0, 1\}$$

- $a_t = 1$: Select item t
- $a_t = 0$: Reject item t

Transition Dynamics

$$x_{t+1}^{(i)} = \begin{cases} a_t \cdot 2 - 1 & \text{if } i = t \text{ (convert 0,1 to -1,1)} \\ x_t^{(i)} & \text{otherwise} \end{cases}$$

$$W_{t+1} = W_t + a_t \cdot w_t$$

$$V_{t+1} = V_t + a_t \cdot v_t$$

4.3 Policy Network Architecture

Item Feature Vector

For item i:

$$\mathbf{f}_i = \left[\frac{v_i}{\bar{v}}, \frac{w_i}{\bar{w}}, \frac{v_i/w_i}{\overline{v/w}}, \frac{w_i}{C_{\max}} \right]$$

Where $\bar{v} = \frac{1}{N}\sum_i v_i, \bar{w} = \frac{1}{N}\sum_i w_i, \overline{v/w} = \frac{1}{N}\sum_i \frac{v_i}{w_i}$.

Item Embedding

$$\mathbf{h}_i = \text{MLP}(\mathbf{f}_i) \in \mathbb{R}^d$$

Context Features

$$\mathbf{h}_{\text{context}} = \left[\frac{W_t}{C_{\max}}, \frac{C_{\max} - W_t}{C_{\max}}, \frac{V_t}{V_{\max}}, \frac{t}{N}, \frac{C_{\max} - W_t}{w_t} \right]$$

Where $V_{\max} = \sum_i v_i$.

The feature $\frac{C_{\max} - W_t}{w_t}$ represents "how many of current item could fit."

Policy Output

$$\mathbf{z}_t = \text{MLP}\left([\mathbf{h}_t \,\|\, \mathbf{h}_{\text{context}}]\right)$$

$$\pi_\theta(a_t = 1 \mid s_t) = \sigma(z_t)$$

Where σ is the sigmoid function.

4.4 Reward Function

Option A: Sparse Terminal Reward

$$r_t = \begin{cases} V_T & \text{if } t = N-1 \text{ and } W_T \le C_{\max} \\ -\lambda(W_T - C_{\max}) & \text{if } t = N-1 \text{ and } W_T > C_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Option B: Dense Reward with Hard Constraint

$$r_t = \begin{cases} v_t & \text{if } a_t = 1 \text{ and } W_t + w_t \le C_{\max} \\ -M & \text{if } a_t = 1 \text{ and } W_t + w_t > C_{\max} \\ 0 & \text{if } a_t = 0 \end{cases}$$

Where M is a large penalty (e.g., $M = V_{\max}$).

Option C: Dense Reward with Soft Constraint (Recommended)

$$r_t = a_t \cdot \left(\alpha \cdot \frac{v_t}{V_{\max}} - \beta \cdot \frac{\max(0, W_t + w_t - C_{\max})}{C_{\max}} \right)$$

Typical values: $\alpha = 1.0, \beta = 2.0$.

Option D: Potential-Based Shaping

Define potential:

$$\Phi(s_t) = \frac{V_t}{V_{\max}} - \gamma_{\text{penalty}} \cdot \frac{\max(0, W_t - C_{\max})}{C_{\max}}$$

Shaped reward:

$$r_t^{\text{shaped}} = r_t + \gamma \Phi(s_{t+1}) - \Phi(s_t)$$

This preserves optimal policy while providing denser signal.

4.5 Value Network

$$V_\phi(s_t) = \text{MLP}\left(\left[\frac{V_t}{V_{\max}}, \frac{W_t}{C_{\max}}, \frac{C_{\max} - W_t}{C_{\max}}, \frac{N - t}{N}, \frac{\sum_{i>t} v_i}{V_{\max}}\right]\right)$$

The last feature represents the "remaining potential value."

4.6 Feasibility Masking (Alternative to Soft Constraints)

Instead of penalizing infeasible actions, mask them:

$$\pi_\theta(a_t = 1 \mid s_t) = \begin{cases} \sigma(z_t) & \text{if } W_t + w_t \le C_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Renormalize:

$$\pi_\theta(a_t \mid s_t) = \frac{\tilde{\pi}_\theta(a_t \mid s_t)}{\sum_{a' \in \{0,1\}} \tilde{\pi}_\theta(a' \mid s_t)}$$

This guarantees feasible solutions.

---

5. Graph Coloring

5.1 Problem Definition

Given an undirected graph $G = (V, E)$ with $|V| = N$ nodes and edge set E, assign colors from $\{1, 2, \ldots, K\}$ to each node such that no two adjacent nodes share the same color:

$$\forall (i, j) \in E : c_i \neq c_j$$

Optimization version: Minimize the number of colors used (chromatic number).

Decision version: Can the graph be colored with K colors?

5.2 MDP Formulation

State Space

$$s_t = (\mathbf{A}, \mathbf{c}_t, t)$$

Where:

- $\mathbf{A} \in \{0,1\}^{N \times N}$: Adjacency matrix, $A_{ij} = 1$ if $(i,j) \in E$
- $\mathbf{c}_t \in \{0, 1, \ldots, K\}^N$: Color assignment, $c_t^{(i)} = 0$ means uncolored
- t: Current step (which node to color)

Action Space

For sequential node coloring (node t at step t):

$$\mathcal{A}(s_t) = \{1, 2, \ldots, K\}$$

Or with constraint (only valid colors):

$$\mathcal{A}_{\text{valid}}(s_t) = \{k \in \{1, \ldots, K\} : \forall j \in \mathcal{N}(t), c_t^{(j)} \neq k\}$$

Where $\mathcal{N}(i) = \{j : A_{ij} = 1\}$ is the neighborhood of node i.

Transition Dynamics

$$c_{t+1}^{(i)} = \begin{cases} a_t & \text{if } i = t \\ c_t^{(i)} & \text{otherwise} \end{cases}$$

5.3 Policy Network Architecture

Node Feature Vector

For node i at step t:

$$\mathbf{f}_i = \left[\text{degree}(i), \mathbf{e}_{c_t^{(i)}}, 1[i = t], \mathbf{b}_i \right]$$

Where:

- $\text{degree}(i) = \sum_j A_{ij}$
- $\mathbf{e}_{c_t^{(i)}} \in \mathbb{R}^{K+1}$: One-hot encoding of current color (0 = uncolored)
- $1[i = t]$: Indicator for current node
- $\mathbf{b}_i \in \{0, 1\}^K$: Blocked colors, $b_i^{(k)} = 1$ if neighbor has color k

$$b_i^{(k)} = 1\left[\exists j \in \mathcal{N}(i) : c_t^{(j)} = k\right]$$

Graph Neural Network (GNN)

Initial embedding:

$$\mathbf{h}_i^{(0)} = \text{Linear}(\mathbf{f}_i)$$

Message passing (for L layers):

$$\mathbf{m}_i^{(l)} = \text{Aggregate}_{j \in \mathcal{N}(i)} \text{MLP}_{\text{msg}}(\mathbf{h}_j^{(l)})$$

$$\mathbf{h}_i^{(l+1)} = \text{LayerNorm}\left(\mathbf{h}_i^{(l)} + \text{MLP}_{\text{update}}\left(\left[\mathbf{h}_i^{(l)} \parallel \mathbf{m}_i^{(l)}\right]\right)\right)$$

Common aggregations:

- Mean: $\text{Aggregate} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)}$
- Sum: $\text{Aggregate} = \sum_{j \in \mathcal{N}(i)}$
- Max: $\text{Aggregate} = \max_{j \in \mathcal{N}(i)}$

Color Embedding

$$\mathbf{e}_k = \text{Embedding}(k) \in \mathbb{R}^d, \quad k \in \{1, \ldots, K\}$$

Policy Output

For current node t:

$$\mathbf{q} = \text{Linear}(\mathbf{h}_t^{(L)})$$

For each color k:

$$u_k = \mathbf{q}^\top \mathbf{e}_k$$

With masking for invalid colors:

$$\tilde{u}_k = \begin{cases} u_k & \text{if } k \in \mathcal{A}_{\text{valid}}(s_t) \\ -\infty & \text{otherwise} \end{cases}$$

Policy:

$$\pi_\theta(a_t = k \mid s_t) = \frac{\exp(\tilde{u}_k)}{\sum_{k'=1}^{K} \exp(\tilde{u}_{k'})}$$

5.4 Reward Function

Definitions

Number of conflicts:

$$C(s) = \sum_{(i,j) \in E} 1[c^{(i)} = c^{(j)} \neq 0]$$

Number of colors used:

$$K_{\text{used}}(s) = |\{c^{(i)} : c^{(i)} \neq 0\}|$$

Number of uncolored nodes:

$$M(s) = \sum_{i=1}^{N} 1[c^{(i)} = 0]$$

Option A: Sparse Terminal Reward

$$r_t = \begin{cases} R_{\text{success}} & \text{if } t = N-1 \text{ and } C(s_T) = 0 \\ -C(s_T) & \text{if } t = N-1 \text{ and } C(s_T) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Option B: Dense Conflict-Based Reward

$$r_t = -\alpha \cdot \Delta C_t$$

Where:

$$\Delta C_t = C(s_{t+1}) - C(s_t) = |\{j \in \mathcal{N}(t) : c_t^{(j)} = a_t\}|$$

This counts how many conflicts are created by coloring node t with color a_t.

Option C: Dense Reward with Color Efficiency (Recommended)

$$r_t = -\alpha \cdot \Delta C_t - \beta \cdot 1[a_t \text{ is a new color}]$$

Where:

$$1[a_t \text{ is a new color}] = 1\left[a_t \notin \{c_t^{(i)} : i < t\}\right]$$

This encourages using fewer colors.

Option D: Multi-Objective Reward

$$r_t = \begin{cases} +1 & \text{if } \Delta C_t = 0 \text{ (valid coloring)} \\ -\alpha \cdot \Delta C_t - \beta \cdot 1[\text{new color}] & \text{otherwise} \end{cases}$$

Typical values: $\alpha = 1.0, \beta = 0.1$.

5.5 Value Network

$$V_\phi(s_t) = \text{MLP}\left(\left[\bar{\mathbf{h}}, \frac{t}{N}, \frac{C(s_t)}{|E|}, \frac{K_{\text{used}}(s_t)}{K}\right]\right)$$

Where $\bar{\mathbf{h}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{h}_i^{(L)}$ is the graph embedding.

5.6 Node Ordering Strategies

The order in which nodes are colored affects solution quality. Options:

1. **Fixed order**: Color nodes $1, 2, \ldots, N$ sequentially
2. **Degree-based**: Color high-degree nodes first (greedy heuristic)

3. **Learned order**: Use another policy to select which node to color next

For learned ordering, modify action space:

$$\mathcal{A}(s_t) = \{(i, k) : c_t^{(i)} = 0, k \in \{1, \ldots, K\}\}$$

---

6. Training Algorithm

6.1 PPO Training Loop

```
Algorithm: PPO for Combinatorial Optimization
──────────────────────────────────────────────────────────────

Input: Initial policy θ, value function ϕ
       Number of iterations M
       Batch size B, epochs per iteration K
       Hyperparameters: γ, λ, ε, c₁, c₂

For iteration = 1, 2, ..., M:

   # Collect trajectories
   For episode = 1, 2, ..., B:
      Sample problem instance
      s₀ ← initial state
      For t = 0, 1, ..., T-1:
         aₜ ~ πθ(·|sₜ)
         rₜ, sₜ₊₁ ← step(sₜ, aₜ)
         Store (sₜ, aₜ, rₜ, πθ(aₜ|sₜ))

   # Compute advantages
   For each trajectory:
      For t = T-1, T-2, ..., 0:
         δₜ = rₜ + γV_ϕ(sₜ₊₁) - V_ϕ(sₜ)
         Âₜ = δₜ + γλÂₜ₊₁  (with ÂT = 0)
      R̂ₜ = Âₜ + V_ϕ(sₜ)

   # Normalize advantages
   Â ← (Â - mean(Â)) / (std(Â) + ε)

   # PPO update
   For epoch = 1, 2, ..., K:
      For mini-batch in shuffle(data):

         # Policy loss
```

ϱ = πθ(als) / πθ_old(als)
L_CLIP = min(ϱÂ, clip(ϱ, 1-ε, 1+ε)Â)

# Value loss
L_VF = (V_ϕ(s) - R̂)²

# Entropy bonus
H = -Σ πθ(als) log πθ(als)

# Total loss
L = -L_CLIP + c_1·L_VF - c_2·H

Update
θ, ϕ ← Adam(θ, ϕ, ∇L)

Output: Trained policy πθ

6.2 Key Implementation Details

Advantage Normalization

$$\hat{A}_t \leftarrow \frac{\hat{A}_t - \mu_{\hat{A}}}{\sigma_{\hat{A}} + \epsilon}$$

Where $\mu_{\hat{A}}$ and $\sigma_{\hat{A}}$ are batch mean and std. This stabilizes training.

Value Function Clipping (Optional)

$$\mathcal{L}^{\text{VF-CLIP}} = \max\left((V_\phi - \hat{R})^2, (\text{clip}(V_\phi, V_{\text{old}} - \epsilon_v, V_{\text{old}} + \epsilon_v) - \hat{R})^2\right)$$

Learning Rate Schedule

Linear decay:

$$\alpha_t = \alpha_0 \cdot \left(1 - \frac{t}{T_{\max}}\right)$$

Gradient Clipping

$$\mathbf{g} \leftarrow \frac{\mathbf{g}}{\max(1, \|\mathbf{g}\|_2 / g_{\max})}$$

Typical $g_{\max} = 0.5$.

6.3 Hyperparameters Summary

Parameter	Symbol	Typical Value	Description
Discount factor	γ	0.99 or 1.0	Use 1.0 for finite episodes
GAE parameter	λ	0.95	Bias-variance trade-off
Clip parameter	ϵ	0.1 - 0.2	Policy update constraint
Value coefficient	c_1	0.5	Value loss weight
Entropy coefficient	c_2	0.01	Exploration bonus
Learning rate	α	1e-4 to 3e-4	Adam optimizer
Batch size	B	64 - 512	Episodes per iteration
Epochs	K	3 - 10	Updates per batch
Gradient clip	$g_{\max}$	0.5	Max gradient norm

---

7. Curriculum Learning

7.1 Motivation

Training directly on hard instances (large N, dense graphs) often fails because:

- Sparse rewards provide little learning signal
- Large action spaces make exploration difficult
- Policy gets stuck in poor local optima

Curriculum learning addresses this by starting with easier instances and gradually increasing difficulty.

7.2 Curriculum Strategies by Problem

TSP Curriculum

Dimension 1: Number of cities

$$N_{\text{curriculum}} = [20, 30, 50, 75, 100, \ldots]$$

Dimension 2: Spatial distribution

- Clustered cities (easier) → Uniform random (harder)

Advancement criterion:

$$\text{Gap}_{\text{avg}} = \frac{L_{\text{policy}} - L_{\text{optimal}}}{L_{\text{optimal}}} < \tau$$

Advance to next level when $\text{Gap}_{\text{avg}} < 5\%$ over 100 instances.

Knapsack Curriculum

Dimension 1: Number of items

$$N_{\text{curriculum}} = [10, 20, 50, 100, \ldots]$$

Dimension 2: Capacity ratio

$$\rho = \frac{C_{\max}}{\sum_i w_i} \in [0.8, 0.6, 0.5, 0.3, \ldots]$$

Higher ratio = easier (more items fit).

Dimension 3: Correlation

- Uncorrelated (values and weights independent) — easier
- Weakly correlated — medium
- Strongly correlated ($v_i \approx w_i$) — harder

Graph Coloring Curriculum

Dimension 1: Number of nodes

$$N_{\text{curriculum}} = [20, 50, 100, 200, \ldots]$$

Dimension 2: Edge density

$$p = \frac{|E|}{\binom{N}{2}} \in [0.1, 0.2, 0.3, 0.5, \ldots]$$

Dimension 3: Number of colors

- $K = \chi(G) + 2$ (easy, extra colors available)
- $K = \chi(G) + 1$ (medium)
- $K = \chi(G)$ (hard, minimum colors)

Where $\chi(G)$ is the chromatic number.

7.3 Curriculum Implementation

Method 1: Discrete Levels

```
level = 0
threshold = 0.9  # success rate

while level < max_level:
    train on instances from level
    if success_rate > threshold:
        level += 1
```

Method 2: Continuous Curriculum (Self-Paced)

Define difficulty score $d(I)$ for instance I. Sample instances with probability:

$$P(I) \propto \exp\left(-\frac{(d(I) - d_{\text{target}})^2}{2\sigma^2}\right)$$

Gradually increase d_{target} based on policy performance.

Method 3: Mixed Curriculum

Sample from multiple difficulty levels:

$$P(\text{level } = k) = \begin{cases} 0.6 & k = \text{current level} \\ 0.2 & k = \text{current level} - 1 \\ 0.2 & k = \text{current level} + 1 \end{cases}$$

This prevents catastrophic forgetting.

7.4 Transfer Learning

When advancing curriculum:

1. **Warm start**: Initialize from previous level's weights
2. **Architecture scaling**: If network size increases, initialize new parameters randomly, keep trained parameters
3. **Fine-tuning**: Lower learning rate when transferring

Summary

Component	TSP	Knapsack	Graph Coloring
State	Visited mask, current city	Item decisions, weight, value	Color assignments, adjacency
Action	Select unvisited city	Accept/reject item	Assign color to node
Encoder	Transformer	MLP	GNN
Reward (dense)	$-d(c_t, a_t)/D_{\text{avg}}$	$\alpha v_t - \beta \cdot \text{overflow}$	$-\alpha \Delta C_t - \beta \cdot 1[\text{new}]$
Episode length	N	N	N
Curriculum dim	City count, distribution	Item count, capacity ratio	Node count, edge density

The PPO objective remains the same across all problems:

$$\mathcal{L}^{\text{PPO}} = -\mathbb{E}\left[\min(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] + c_1 \mathcal{L}^{\text{VF}} - c_2 \mathcal{H}$$

References

1. Schulman, J., et al. "Proximal Policy Optimization Algorithms." arXiv:1707.06347 (2017)
2. Kool, W., et al. "Attention, Learn to Solve Routing Problems!" ICLR (2019)
3. Bello, I., et al. "Neural Combinatorial Optimization with Reinforcement Learning." arXiv:1611.09940 (2016)
4. Bengio, Y., et al. "Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon." EJOR (2021)
5. Mazyavkina, N., et al. "Reinforcement Learning for Combinatorial Optimization: A Survey." Computers & OR (2021)