Appendix: Mathematical Formulation of Proximal Policy Optimization for Combinatorial Optimization Problems

---

A. Preliminaries

A.1 Markov Decision Process

We formulate combinatorial optimization problems as finite-horizon Markov Decision Processes (MDPs). An MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, T)$, where:

- $\mathcal{S}$ denotes the state space,
- $\mathcal{A}$ denotes the action space,
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the transition probability function,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function,
- $\gamma \in [0, 1]$ denotes the discount factor,
- $T \in \mathbb{N}$ denotes the horizon (episode length).

A trajectory τ is a sequence of states and actions:

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$

where $r_t = \mathcal{R}(s_t, a_t)$ denotes the reward received at time step t.

A.2 Policy

A stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines a probability distribution over actions conditioned on states, parameterized by $\theta \in \mathbb{R}^d$:

$$\pi_\theta(a \mid s) = \mathbb{P}(a_t = a \mid s_t = s; \theta)$$

The policy satisfies the probability axioms:

$$\sum_{a \in \mathcal{A}(s)} \pi_\theta(a \mid s) = 1, \quad \forall s \in \mathcal{S}$$

where $\mathcal{A}(s) \subseteq \mathcal{A}$ denotes the set of valid actions in state s.

A.3 Value Functions

Definition A.1 (State Value Function). The state value function $V^\pi : \mathcal{S} \to \mathbb{R}$ under policy π is defined as the expected cumulative discounted reward starting from state s:

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t r_t \,\middle|\, s_0 = s \right]$$

where the expectation is taken over trajectories τ generated by following policy π.

Definition A.2 (Action Value Function). The action value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ under policy π is defined as the expected cumulative discounted reward starting from state s, taking action a, and thereafter following policy π:

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a \right]$$

Definition A.3 (Advantage Function). The advantage function $A^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ quantifies the relative benefit of taking action a in state s compared to the average action under policy π:

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

Remark A.1. The advantage function satisfies $\mathbb{E}_{a \sim \pi(\cdot|s)}[A^\pi(s, a)] = 0$ for all states $s \in \mathcal{S}$.

A.4 Bellman Equations

The value functions satisfy the Bellman equations:

Bellman Equation for V^π:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}[V^\pi(s')] \right]$$

Bellman Equation for Q^π:

$$Q^\pi(s, a) = \mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} \left[\mathbb{E}_{a' \sim \pi(\cdot|s')}[Q^\pi(s', a')] \right]$$

B. Policy Gradient Methods

B.1 Objective Function

The objective of reinforcement learning is to find a policy that maximizes the expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right] = \mathbb{E}_{s_0 \sim \rho_0}[V^{\pi_\theta}(s_0)]$$

where ρ_0 denotes the initial state distribution.

B.2 Policy Gradient Theorem

Theorem B.1 (Policy Gradient Theorem). The gradient of the objective function with respect to the policy parameters is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot A^{\pi_\theta}(s_t, a_t)\right]$$

Proof. See Sutton et al. (2000) for the complete derivation. □

B.3 REINFORCE Estimator

The policy gradient can be estimated using Monte Carlo sampling:

$$\hat{g} = \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \hat{A}_t$$

where $\mathcal{B}$ denotes a batch of trajectories and $\hat{A}_t$ denotes an estimator of the advantage function.

C. Proximal Policy Optimization

C.1 Trust Region Methods

Trust region methods constrain the policy update to prevent destructively large changes. The Trust Region Policy Optimization (TRPO) objective is:

$$\max_\theta \quad \mathbb{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t\right]$$

$$\text{subject to} \quad \mathbb{E}_t\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t))\right] \le \delta$$

where D_{KL} denotes the Kullback-Leibler divergence.

C.2 PPO Clipped Objective

Proximal Policy Optimization (PPO) replaces the hard constraint with a clipped objective function.

Definition C.1 (Probability Ratio). The probability ratio between the current and old policies is defined as:

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

Definition C.2 (Clipped Surrogate Objective). The PPO clipped objective is defined as:

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

where $\epsilon \in (0,1)$ is the clipping hyperparameter and the clip function is defined as:

$$\text{clip}(x, a, b) = \max(a, \min(x, b))$$

Proposition C.1. The clipped objective provides a lower bound on the unclipped objective when $\hat{A}_t > 0$ and the ratio ρ_t exceeds $1+\epsilon$, and similarly when $\hat{A}_t < 0$ and ρ_t falls below $1-\epsilon$.

Proof. Consider two cases:

Case 1: $\hat{A}_t \geq 0$ (advantageous action).

- If $\rho_t \leq 1+\epsilon$: $\min(\rho_t\hat{A}_t, (1+\epsilon)\hat{A}_t) = \rho_t\hat{A}_t$
- If $\rho_t > 1+\epsilon$: $\min(\rho_t\hat{A}_t, (1+\epsilon)\hat{A}_t) = (1+\epsilon)\hat{A}_t < \rho_t\hat{A}_t$

Case 2: $\hat{A}_t < 0$ (disadvantageous action).

- If $\rho_t \geq 1-\epsilon$: $\min(\rho_t\hat{A}_t, (1-\epsilon)\hat{A}_t) = \rho_t\hat{A}_t$
- If $\rho_t < 1-\epsilon$: $\min(\rho_t\hat{A}_t, (1-\epsilon)\hat{A}_t) = (1-\epsilon)\hat{A}_t < \rho_t\hat{A}_t$

Thus, the clipped objective removes incentives for moving the ratio outside $[1-\epsilon, 1+\epsilon]$. $\square$

C.3 Value Function Loss

The value function is trained to minimize the mean squared error between predicted values and empirical returns:

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

where $\hat{R}_t$ denotes the target return and ϕ denotes the value function parameters.

Definition C.3 (Discounted Return). The discounted return from time step t is defined as:

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

C.4 Entropy Regularization

To encourage exploration and prevent premature convergence, an entropy bonus is added to the objective.

Definition C.4 (Policy Entropy). The entropy of the policy distribution at state s is:

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

The entropy loss is defined as:

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

C.5 Combined PPO Objective

The complete PPO objective function is:

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

where $c_1, c_2 > 0$ are weighting coefficients.

Remark C.1. The negative sign before $\mathcal{L}^{\text{CLIP}}$ converts the maximization problem to minimization. Similarly, the positive sign before $\mathcal{L}^{\text{ENT}}$ encourages higher entropy (since $\mathcal{L}^{\text{ENT}}$ is the negative entropy).

D. Generalized Advantage Estimation

D.1 Temporal Difference Residual

Definition D.1 (TD Residual). The temporal difference residual at time step t is defined as:

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

where $V_\phi(s_T) = 0$ for terminal states.

Proposition D.1. The TD residual is an unbiased estimator of the advantage function when the value function is exact, i.e., $V_\phi = V^\pi$:

$$\mathbb{E}[\delta_t \mid s_t, a_t] = A^\pi(s_t, a_t)$$

D.2 GAE Definition

Definition D.2 (Generalized Advantage Estimation). The GAE estimator with parameters (γ, λ) is defined as:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{k=0}^{T-1-t} (\gamma\lambda)^k \delta_{t+k}$$

Expanding this expression:

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

D.3 Recursive Formulation

Proposition D.2. The GAE estimator satisfies the following recursive relationship:

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1}$$

with boundary condition $\hat{A}_T = 0$.

Proof. By definition:

$$\hat{A}_t = \sum_{k=0}^{T-1-t} (\gamma\lambda)^k \delta_{t+k} = \delta_t + \gamma\lambda \sum_{k=0}^{T-2-t} (\gamma\lambda)^k \delta_{t+1+k} = \delta_t + \gamma\lambda\hat{A}_{t+1}$$

□

D.4 Bias-Variance Trade-off

Proposition D.3. The GAE parameter λ controls the bias-variance trade-off:

- When $\lambda = 0$: $\hat{A}_t^{\mathrm{GAE}(\gamma,0)} = \delta_t$ (one-step TD, low variance, high bias)
- When $\lambda = 1$: $\hat{A}_t^{\mathrm{GAE}(\gamma,1)} = \sum_{k=0}^{T-1-t} \gamma^k r_{t+k} - V_\phi(s_t)$ (Monte Carlo, high variance, low bias)

Proof. For $\lambda = 0$:

$$\hat{A}_t^{\text{GAE}(\gamma,0)} = \sum_{k=0}^{T-1-t} 0^k \delta_{t+k} = \delta_t$$

For $\lambda = 1$:

$$\hat{A}_t^{\text{GAE}(\gamma,1)} = \sum_{k=0}^{T-1-t} \gamma^k \delta_{t+k} = \sum_{k=0}^{T-1-t} \gamma^k (r_{t+k} + \gamma V_\phi(s_{t+k+1}) - V_\phi(s_{t+k}))$$

The telescoping sum yields:

$$\hat{A}_t^{\text{GAE}(\gamma,1)} = \sum_{k=0}^{T-1-t} \gamma^k r_{t+k} + \gamma^{T-t} V_\phi(s_T) - V_\phi(s_t) = \sum_{k=0}^{T-1-t} \gamma^k r_{t+k} - V_\phi(s_t)$$

since $V_\phi(s_T) = 0$. □

D.5 Advantage Normalization

To reduce variance and stabilize training, advantages are normalized across each batch:

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

where $\mu_{\hat{A}} = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \hat{A}_t$, $\sigma_{\hat{A}} = \sqrt{\frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} (\hat{A}_t - \mu_{\hat{A}})^2}$, and $\varepsilon > 0$ is a small constant for numerical stability.

E. Problem Formulations

E.1 Traveling Salesman Problem

E.1.1 Problem Definition

Definition E.1 (Traveling Salesman Problem). Given a set of N cities with coordinates $\mathbf{P} = \{p_1, p_2, \ldots, p_N\}$ where $p_i \in \mathbb{R}^2$, the Traveling Salesman Problem (TSP) seeks a permutation $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$ of $\{1, 2, \ldots, N\}$ that minimizes the total tour length:

$$L(\tau) = \sum_{i=1}^{N-1} d(p_{\tau_i}, p_{\tau_{i+1}}) + d(p_{\tau_N}, p_{\tau_1})$$

where $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ denotes the Euclidean distance:

$$d(p_i, p_j) = \|p_i - p_j\|_2 = \sqrt{(p_i^{(x)} - p_j^{(x)})^2 + (p_i^{(y)} - p_j^{(y)})^2}$$

E.1.2 MDP Formulation

State Space. At time step $t \in \{0, 1, \ldots, N-1\}$, the state is defined as:

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

where:

- $\mathbf{P} \in \mathbb{R}^{N \times 2}$: City coordinate matrix (static)
- $\mathbf{v}_t \in \{0, 1\}^N$: Visitation indicator vector, $v_t^{(i)} = 1[\text{city } i \text{ visited by time } t]$
- $c_t \in \{1, \ldots, N\}$: Index of current city
- $c_0 \in \{1, \ldots, N\}$: Index of starting city (depot)

Action Space. The action space at state s_t consists of all unvisited cities:

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

Transition Function. The transition function is deterministic:

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

with updates:

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

Episode Termination. The episode terminates when all cities are visited, i.e., when $\sum_{i=1}^{N} v_t^{(i)} = N$.

E.1.3 Reward Function

Definition E.2 (TSP Reward Function). We define the dense reward function as:

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

where D_{avg} is the average pairwise distance:

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

At the terminal step $t = N - 1$, the reward includes the return distance:

$$r_{N-1} = -\frac{d(p_{c_{N-1}}, p_{a_{N-1}}) + d(p_{a_{N-1}}, p_{c_0})}{D_{\text{avg}}}$$

Remark E.1. The normalization by D_{avg} ensures that rewards are scale-invariant across problem instances of varying sizes.

E.1.4 Policy Architecture

Node Embedding. Each city i is embedded as:

$$\mathbf{h}_i^{(0)} = \mathbf{W}_{\text{emb}} \cdot p_i + \mathbf{b}_{\text{emb}} \in \mathbb{R}^{d_h}$$

Transformer Encoder. For layers $l = 0, 1, \ldots, L-1$:

Multi-Head Attention:

$$\text{MHA}(\mathbf{H}^{(l)}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O$$

where each head is computed as:

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

with $\mathbf{Q}_j = \mathbf{H}^{(l)}\mathbf{W}_j^Q, \mathbf{K}_j = \mathbf{H}^{(l)}\mathbf{W}_j^K, \mathbf{V}_j = \mathbf{H}^{(l)}\mathbf{W}_j^V$.

Layer update with residual connections:

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

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

Context Vector. The decoder context combines global and local information:

$$\mathbf{h}_{\text{ctx}} = \left[\bar{\mathbf{h}} \parallel \mathbf{h}_{c_t}^{(L)} \parallel \mathbf{h}_{c_0}^{(L)}\right] \in \mathbb{R}^{3d_h}$$

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

Action Probabilities. The policy computes attention scores:

$$u_i = \begin{cases} C \cdot \tanh\left(\frac{(\mathbf{W}_q \mathbf{h}_{\text{ctx}})^\top (\mathbf{W}_k \mathbf{h}_i^{(L)})}{\sqrt{d_k}}\right) & \text{if } v_t^{(i)} = 0 \\ -\infty & \text{if } v_t^{(i)} = 1 \end{cases}$$

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

where $C > 0$ is a clipping constant (typically $C = 10$).

E.1.5 Value Function

The value network estimates the expected future reward:

$$V_\phi(s_t) = \text{MLP}\left(\bar{\mathbf{h}} \parallel \mathbf{h}_{c_t}^{(L)} \parallel \frac{t}{N}\right)$$

E.2 Knapsack Problem

E.2.1 Problem Definition

Definition E.3 (0-1 Knapsack Problem). Given N items, where item i has value $v_i > 0$ and weight $w_i > 0$, and a knapsack with capacity $C_{\max} > 0$, the 0-1 Knapsack Problem seeks a binary selection vector $\mathbf{x}^* \in \{0,1\}^N$ that maximizes total value subject to the capacity constraint:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\{0,1\}^N} \sum_{i=1}^{N} v_i x_i \quad \text{subject to} \quad \sum_{i=1}^{N} w_i x_i \leq C_{\max}$$

E.2.2 MDP Formulation

State Space. At time step $t \in \{0, 1, \ldots, N-1\}$, the state is defined as:

$$s_t = (\mathbf{v}, \mathbf{w}, \mathbf{x}_t, W_t, V_t, C_{\text{max}}) \in \mathcal{S}$$

where:

- $\mathbf{v} \in \mathbb{R}^N_{>0}$: Item value vector (static)
- $\mathbf{w} \in \mathbb{R}^N_{>0}$: Item weight vector (static)
- $\mathbf{x}_t \in \{-1, 0, 1\}^N$: Decision vector ($1$: selected, -1: rejected, 0: undecided)
- $W_t = \sum_{i:x_t^{(i)}=1} w_i$: Accumulated weight
- $V_t = \sum_{i:x_t^{(i)}=1} v_i$: Accumulated value
- C_{max}: Knapsack capacity (static)

Action Space. For sequential item processing:

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

where $a_t = 1$ indicates selecting item t and $a_t = 0$ indicates rejecting item t.

Transition Function. The deterministic transition updates:

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

$$W_{t+1} = W_t + a_t \cdot w_t, \quad V_{t+1} = V_t + a_t \cdot v_t$$

E.2.3 Reward Function

Definition E.4 (Knapsack Reward Function). The dense reward function with soft constraints is:

$$r_t = \mathcal{R}(s_t, a_t) = a_t \cdot \left(\alpha \cdot \frac{v_t}{V_{\text{sum}}} - \beta \cdot \frac{O_t}{C_{\text{max}}} \right)$$

where:

- $V_{\text{sum}} = \sum_{i=1}^N v_i$ is the total value of all items
- $O_t = \max(0, W_t + w_t - C_{\text{max}})$ is the overflow (constraint violation)
- $\alpha, \beta > 0$ are weighting coefficients

Proposition E.1. Setting $\beta > \alpha \cdot \max_i \frac{v_i/V_{\text{sum}}}{w_i/C_{\max}}$ ensures that the penalty for overflow exceeds the reward for any single item.

E.2.4 Policy Architecture

Item Features. For item i:

$$\mathbf{f}_i = \left[\frac{v_i}{\bar{v}}, \frac{w_i}{\bar{w}}, \frac{v_i/w_i}{\overline{(v/w)}}, \frac{w_i}{C_{\max}}, x_t^{(i)}\right]^\top \in \mathbb{R}^5$$

Context Features. Global state information:

$$\mathbf{f}_{\text{ctx}} = \left[\frac{W_t}{C_{\max}}, \frac{C_{\max} - W_t}{C_{\max}}, \frac{V_t}{V_{\text{sum}}}, \frac{t}{N}, \frac{\min(1, (C_{\max} - W_t)/w_t)}{1}\right]^\top \in \mathbb{R}^5$$

Policy Network. The policy outputs the probability of selection:

$$\mathbf{z}_t = \text{MLP}\left(\text{MLP}_{\text{item}}(\mathbf{f}_t) \,\|\, \mathbf{f}_{\text{ctx}}\right)$$

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

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function.

E.2.5 Feasibility Masking

Definition E.5 (Masked Policy). To guarantee feasible solutions, the policy is masked:

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

The normalized policy is:

$$\pi_\theta(a_t \mid s_t) = \frac{\tilde{\pi}_\theta(a_t \mid s_t)}{\tilde{\pi}_\theta(0 \mid s_t) + \tilde{\pi}_\theta(1 \mid s_t)}$$

E.3 Graph Coloring Problem

E.3.1 Problem Definition

Definition E.6 (Graph Coloring Problem). Given an undirected graph $G = (V, E)$ with vertex set $V = \{1, 2, \ldots, N\}$ and edge set $E \subseteq \binom{V}{2}$, and a set of K colors $\mathcal{C} = \{1, 2, \ldots, K\}$, the Graph Coloring Problem seeks a color assignment $\mathbf{c} : V \to \mathcal{C}$ such that no two adjacent vertices share the same color:

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

Definition E.7 (Chromatic Number). The chromatic number $\chi(G)$ is the minimum number of colors required for a valid coloring:

$$\chi(G) = \min\{K : \exists \text{ valid } K\text{-coloring of } G\}$$

E.3.2 MDP Formulation

State Space. At time step $t \in \{0, 1, \ldots, N - 1\}$, the state is defined as:

$$s_t = (\mathbf{A}, \mathbf{c}_t) \in \mathcal{S}$$

where:

- $\mathbf{A} \in \{0, 1\}^{N \times N}$: Adjacency matrix, $A_{ij} = 1[(i, j) \in E]$
- $\mathbf{c}_t \in \{0, 1, \ldots, K\}^N$: Color assignment vector, $c_t^{(i)} = 0$ indicates uncolored

Neighborhood. The neighborhood of vertex i is defined as:

$$\mathcal{N}(i) = \{j \in V : A_{ij} = 1\}$$

Action Space. For coloring vertex t:

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

Valid Action Space. Colors not used by neighbors:

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

Transition Function. The deterministic transition updates:

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

E.3.3 Reward Function

Definition E.8 (Conflict Count). The number of conflicts in state s is:

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

Definition E.9 (Graph Coloring Reward Function). The dense reward function is:

$$r_t = \mathcal{R}(s_t, a_t) = -\alpha \cdot \Delta C_t - \beta \cdot 1[a_t \notin \mathcal{C}_{\text{used}}(s_t)]$$

where:

- $\Delta C_t = C(s_{t+1}) - C(s_t) = |\{j \in \mathcal{N}(t) : c_t^{(j)} = a_t\}|$ is the number of new conflicts
- $\mathcal{C}_{\text{used}}(s_t) = \{c_t^{(i)} : c_t^{(i)} \neq 0\}$ is the set of colors already used
- $\alpha, \beta > 0$ are weighting coefficients

Remark E.2. The term $1[a_t \notin \mathcal{C}_{\text{used}}(s_t)]$ penalizes introducing new colors, encouraging solutions with fewer distinct colors.

E.3.4 Policy Architecture

Node Features. For vertex i at step t:

$$\mathbf{f}_i = \left[\frac{\deg(i)}{\Delta(G)}, \mathbf{e}_{c_t^{(i)}}, 1[i = t], \mathbf{b}_i\right]^\top$$

where:

- $\deg(i) = |\mathcal{N}(i)|$ is the degree of vertex i
- $\Delta(G) = \max_i \deg(i)$ is the maximum degree
- $\mathbf{e}_k \in \{0,1\}^{K+1}$ is the one-hot encoding of color k
- $\mathbf{b}_i \in \{0,1\}^K$ is the blocked color vector: $b_i^{(k)} = 1[\exists j \in \mathcal{N}(i) : c_t^{(j)} = k]$

Graph Neural Network. Message passing for L layers:

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

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

where AGG denotes an aggregation function (mean, sum, or max).

Color Embedding. Each color k is embedded as:

$$\mathbf{e}_k = \text{Embedding}(k) \in \mathbb{R}^{d_h}$$

Action Probabilities. For current vertex t:

$$u_k = \begin{cases} (\mathbf{W}_q \mathbf{h}_t^{(L)})^\top \mathbf{e}_k & \text{if } k \in \mathcal{A}_{\text{valid}}(s_t) \\ -\infty & \text{otherwise} \end{cases}$$

$$\pi_\theta(a_t = k \mid s_t) = \frac{\exp(u_k)}{\sum_{k'=1}^{K} \exp(u_{k'})}$$

F. Training Procedure

F.1 Algorithm

Algorithm 1: PPO for Combinatorial Optimization

Input: Initial policy parameters θ, value function parameters ϕ, number of iterations M, batch size B, number of epochs K, hyperparameters $\gamma, \lambda, \epsilon, c_1, c_2$

Output: Trained policy parameters θ^*

---

for iteration $= 1, 2, \ldots, M$ **do**

  Initialize trajectory buffer $\mathcal{D} \leftarrow \emptyset$

  for episode $= 1, 2, \ldots, B$ **do**

    Sample problem instance $\mathcal{I}$

    Initialize state $s_0 \leftarrow \text{InitState}(\mathcal{I})$

    for $t = 0, 1, \ldots, T-1$ **do**

      Sample action $a_t \sim \pi_\theta(\cdot \mid s_t)$

Compute reward $r_t \leftarrow \mathcal{R}(s_t, a_t)$

Compute next state $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$

Store $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, \pi_\theta(a_t \mid s_t))\}$

end for

end for

Compute advantages $\{\hat{A}_t\}$ using GAE (Definition D.2)

Compute returns $\{\hat{R}_t\}$ where $\hat{R}_t = \hat{A}_t + V_\phi(s_t)$

Normalize advantages: $\hat{A}_t \leftarrow (\hat{A}_t - \mu_{\hat{A}})/(\sigma_{\hat{A}} + \varepsilon)$

for epoch $= 1, 2, \ldots, K$ **do**

for mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**

Compute ratio: $\rho_t \leftarrow \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$

Compute clipped objective: $\mathcal{L}^{\text{CLIP}} \leftarrow \frac{1}{|\mathcal{B}|}\sum_{t \in \mathcal{B}} \min(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t)$

Compute value loss: $\mathcal{L}^{\text{VF}} \leftarrow \frac{1}{|\mathcal{B}|}\sum_{t \in \mathcal{B}}(V_\phi(s_t) - \hat{R}_t)^2$

Compute entropy: $\mathcal{H} \leftarrow \frac{1}{|\mathcal{B}|}\sum_{t \in \mathcal{B}} \mathcal{H}[\pi_\theta(\cdot \mid s_t)]$

Compute total loss: $\mathcal{L} \leftarrow -\mathcal{L}^{\text{CLIP}} + c_1 \mathcal{L}^{\text{VF}} - c_2 \mathcal{H}$

Update parameters: $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla_{(\theta,\phi)} \mathcal{L}$

end for

end for

$\theta_{\text{old}} \leftarrow \theta$

end for

return θ

F.2 Hyperparameters

Table 1: Recommended Hyperparameter Values

Symbol	Parameter	Recommended Value
γ	Discount factor	0.99 or 1.0
λ	GAE parameter	0.95
ϵ	Clipping parameter	0.1 to 0.2
c_1	Value loss coefficient	0.5
c_2	Entropy coefficient	0.01
η	Learning rate	10^{-4} to 3×10^{-4}
B	Batch size (episodes)	64 to 512
K	Epochs per iteration	3 to 10
$g_{\max}$	Gradient clipping norm	0.5

G. Curriculum Learning

G.1 Motivation

Definition G.1 (Curriculum Learning). Curriculum learning is a training strategy that presents examples to the model in a meaningful order, typically from easy to hard, to improve learning efficiency and final performance.

G.2 Difficulty Metrics

Definition G.2 (Instance Difficulty). For each problem, we define a difficulty function $\mathcal{D} : \mathcal{I} \to \mathbb{R}_{\geq 0}$:

TSP:

$$\mathcal{D}_{\text{TSP}}(\mathcal{I}) = N$$

Knapsack:

$$\mathcal{D}_{\text{KP}}(\mathcal{I}) = N \cdot \left(1 - \frac{C_{\max}}{\sum_{i=1}^{N} w_i}\right)$$

Graph Coloring:

$$\mathcal{D}_{\text{GC}}(\mathcal{I}) = N \cdot \frac{|E|}{\binom{N}{2}} \cdot \frac{\chi(G)}{K}$$

G.3 Curriculum Schedule

Definition G.3 (Curriculum Schedule). A curriculum schedule $\mathcal{C} : \mathbb{N} \to \mathcal{P}(\mathcal{I})$ maps training iterations to probability distributions over instances.

Progressive Curriculum:

$$\mathcal{D}_{\text{target}}(t) = \mathcal{D}_{\min} + (\mathcal{D}_{\max} - \mathcal{D}_{\min}) \cdot \min\left(1, \frac{t}{T_{\text{curriculum}}}\right)$$

Sampling Distribution:

$$P(\mathcal{I}) \propto \exp\left(-\frac{(\mathcal{D}(\mathcal{I}) - \mathcal{D}_{\text{target}})^2}{2\sigma^2}\right)$$

G.4 Advancement Criteria

Performance-Based Advancement. Advance to the next difficulty level when:

$$\frac{1}{|\mathcal{V}|} \sum_{\mathcal{I} \in \mathcal{V}} 1[\text{success}(\mathcal{I})] \geq \tau_{\text{success}}$$

where $\mathcal{V}$ is a validation set and τ_{success} is the success threshold (typically 0.9).

H. Summary

This appendix presented a complete mathematical formulation of Proximal Policy Optimization (PPO) applied to three NP-hard combinatorial optimization problems: the Traveling Salesman Problem, the 0-1 Knapsack Problem, and the Graph Coloring Problem.

Table 2: Problem Formulation Summary

Component	TSP	Knapsack	Graph Coloring
State dimension	$O(N \times d)$	$O(N)$	$O(N \times d)$
Action space size	$	\mathcal{A}	\leq N$
Episode length	$T = N$	$T = N$	$T = N$
Encoder architecture	Transformer	MLP	GNN
Primary reward signal	Negative distance	Item value	Negative conflicts

The PPO algorithm provides a stable and sample-efficient approach to learning construction heuristics for these problems, with the flexibility to incorporate domain-specific knowledge through reward shaping and architectural design.

References

1. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

2. Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*.

3. Kool, W., van Hoof, H., & Welling, M. (2019). Attention, Learn to Solve Routing Problems! *International Conference on Learning Representations*.

4. Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural Combinatorial Optimization with Reinforcement Learning. *arXiv preprint arXiv:1611.09940*.

5. Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon. *European Journal of Operational Research*, 290(2), 405-421.

6. Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 12.