UCPO: A Universal Constrained Combinatorial Optimization Method via Preference Optimization

Zhanhong Fang[1*], Debing Wang[1*], Jinbiao Chen[2], Jiahai Wang[1], Zizhen Zhang[1†]

[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
[2]Department of Industrial Systems Engineering & Management, National University of Singapore, Singapore
zhangzzh7@mail.sysu.edu.cn

Abstract

Neural solvers have demonstrated remarkable success in combinatorial optimization, often achieving better average quality within the same time limit in solution quality, and generalization. However, their efficacy deteriorates significantly when confronted with complex constraints that cannot be effectively managed through simple masking mechanisms. To address this limitation, we introduce Universal Constrained Preference Optimization (UCPO), a novel plug-and-play framework that seamlessly integrates preference learning into existing neural solvers via a specially designed loss function, without requiring architectural modifications. UCPO embeds constraint satisfaction directly into a preference-based objective, eliminating the need for meticulous hyperparameter tuning. Leveraging a lightweight warm-start fine-tuning protocol, UCPO enables pre-trained models to consistently produce near-optimal, feasible solutions on challenging constraint-laden tasks, achieving exceptional performance with as little as 1% of the original training budget.

Code — https://github.com/Birdie-Go/UCPO
Extended version — https://arxiv.org/abs/2511.10148

Introduction

Combinatorial optimization forms the backbone of numerous mission-critical applications across modern industry and scientific computing. Despite the exponential complexity of their search spaces, these tasks demand near-optimal solutions within practical timeframes. Recent advancements in neural solvers, particularly those grounded in deep reinforcement learning (DRL), have demonstrated significant superiority over classical heuristics in terms of solution quality, computational efficiency, and cross-domain generalization. These solvers have achieved remarkable performance on canonical benchmarks such as the Traveling Salesman Problem (TSP), Capacitated Vehicle Routing Problem (CVRP), and Job-Shop Scheduling Problem (JSP). However, a critical challenge arises when deploying these methods on real-world instances with complex constraints that cannot be effectively managed through simple masking mechanisms.

In practical settings, hard constraints, e.g., non-negotiable customer time windows in vehicle routing (Liu et al. 2023) or draft-depth limitations in port logistics (Fadda et al. 2023), may pose significant difficulties. Traditional neural solvers often struggle to balance feasibility and optimality under such constraints. Existing approaches either rely on intricate masking strategies (Chen et al. 2024), which become computationally infeasible as the decision horizon grows, or employ Lagrangian penalty functions (Tang et al. 2022; Bi et al. 2024), which transform the original single-objective problem into a bi-objective trade-off. The resulting Lagrange multipliers are notoriously sensitive to calibration, often leading to solutions that are either infeasible or suboptimal. This sensitivity necessitates careful, problem-specific tuning, which is both time-consuming and limits the universal applicability of these methods.

To address these limitations, we introduce **Universal Constrained Preference Optimization (UCPO)**, a novel framework that leverages preference optimization (Meng, Xia, and Chen 2024; Pan et al. 2025; Liao et al. 2025) to handle hard constraints in combinatorial optimization tasks.

Our framework introduces the following key innovations:

1. **Seamless Architectural Integration**: UCPO can be effortlessly attached to existing neural combinatorial optimization (NCO) models without altering the underlying architecture. We design a novel universal constrained preference loss function, which includes feasibility margin loss, primal refinement loss, and dual exploration loss. This approach preserves the original model's learned representations while significantly enhancing its ability to handle complex constraints.
2. **Mask-Agnostic Constraint Enforcement**: Unlike traditional methods that rely on progressively refined or multi-step masks to ensure feasibility, UCPO employs a unified partial-order criterion. This criterion prioritizes feasible solutions over infeasible ones and favors infeasible solutions with lower aggregate constraint violations. It also eliminates the need for hand-tuned or adaptively updated Lagrange multipliers of the relaxation problem.
3. **Efficient Warm-Start Adaptation**: Utilizing a pre-trained checkpoint as a high-quality initializer, UCPO

[*]These authors contributed equally.
[†]Corresponding author.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

performs lightweight fine-tuning through pairwise preference sampling and gradient accumulation. This protocol significantly reduces computational resource requirements and enables rapid adaptation to new constraint sets without retraining from scratch.

We conducted extensive experiments across four representative constrained benchmarks: Traveling Salesman Problem with Time Windows (TSPTW), Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), Traveling Salesman Problem with Draft Limit (TSPDL), and Capacitated Vehicle Routing Problem with Time Windows and Limited Vehicles (CVRPTWLV). Each benchmark involves intricate constraints that are notoriously difficult to handle, such as time-windows and resource limitations. UCPO demonstrates exceptional performance compared to state-of-the-art neural methods, establishing it as a general, reusable, and highly efficient paradigm for handling hard constraints in neural combinatorial optimization.

Related Works

Neural solvers for combinatorial optimization can be broadly categorized into three principal classes: end-to-end constructive methods, iterative improvement methods, and non-autoregressive methods. This work primarily focuses on end-to-end constructive methods.

End-to-end constructive methods. Initiated by Pointer Networks (Vinyals, Fortunato, and Jaitly 2015), this category generates solutions step-by-step using autoregressive decoding. Subsequent works such as the Attention Model (AM) (Kool, van Hoof, and Welling 2019) and H-TSP (Pan et al. 2023) enhance policy expressiveness through reinforcement learning and hierarchical decoding. POMO (Kwon et al. 2020) achieves a significant performance leap in small-scale TSP by leveraging solution space symmetries to improve sample efficiency. SymNCO (Kim, Park, and Park 2022) adopts a similar strategy. MatNet (Kwon et al. 2021), ELG (Gao et al. 2024), and PolyNet (Hottung, Mahajan, and Tierney 2025) further improve accuracy for small instances through matrix distance calculations, local policy optimization, and solution diversity mechanisms. To handle large-scale problems, LEHD (Luo et al. 2023) and InViT (Fang et al. 2024) enhance representation capacity and decoding efficiency. GOAL (Drakulic, Michel, and Andreoli 2025), MTPOMO (Liu et al. 2024), and MVMoE (Zhou et al. 2024) pursue multi-task generalization. Notably, BOPO (Liao et al. 2025) and POCO (Pan et al. 2025) introduce preference-optimization concepts, offering novel policy alignment strategies. However, these methods primarily rely on lightweight masking mechanisms, necessitating ad-hoc penalty functions or masking logics when handling additional constraints, thereby limiting their universality.

Other paradigms. Iterative improvement methods (e.g., DACT (Ma et al. 2021), NeuOpt (Ma, Cao, and Chee 2023)) improve solutions via refinement loops, while non-autoregressive methods (e.g., DeepACO (Ye et al. 2023), DIFUSCO (Sun and Yang 2023)) aim to enhance inference efficiency through ant colony or diffusion-based generation. Notably, these paradigms also struggle with global constraints under the same masking limitations.

When constraints are global, non-linear, or both, lightweight masking is often insufficient. Existing work tackles this via two main strategies.

Lagrangian relaxation and penalty functions. A common strategy involves augmenting the objective function with soft constraint penalties or employing two-stage training to mitigate constraint violations (Tang et al. 2022; Chen et al. 2022; Bi et al. 2024). For instance, MUSLA (Chen et al. 2024) incorporates multi-step look-ahead features but is coupled to problem-specific characteristics, limiting its flexibility. PIP (Bi et al. 2024) combines Lagrange multipliers with one-step look-ahead masks to reduce infeasibility rates but struggles to balance feasibility and solution quality. These methods require careful tuning of Lagrange multipliers, which is sensitive to problem-specific parameters and may lead to suboptimal solutions.

Exploration of feasible and infeasible regions. NeuOpt (Ma, Cao, and Chee 2023) guides exploration into infeasible regions followed by k-opt iterative repair. NCG (Xia and Zhang 2024) adopts a column-generation paradigm to predict the probability of constraint satisfaction for each column. Both approaches require increased exploration budgets or the training of auxiliary probabilistic models, adding to the computational and implementation complexities.

In summary, existing neural-based methods for combinatorial optimization face challenges in handling complex constraints, often requiring problem-specific tuning and complex masking strategies. This motivates the need for a versatile, mask-free, and low-hyperparameter solution compatible with general-purpose neural solvers. Our UCPO framework addresses these limitations by offering an efficient and universal approach to constrained combinatorial optimization.

Preliminaries

Lagrangian Methods

Consider a generic combinatorial optimization problem of the form:

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & f(x) \\ \text{subject to} \quad & g_j(x) \le 0, \quad j \in \mathcal{J}, \\ & h_k(x) = 0, \quad k \in \mathcal{K}, \end{aligned}$$

where $x = (x_1, \ldots, x_n)^\top$ represents the decision variables, $\mathcal{X} \subseteq \mathbb{R}^n$ defines the feasible domain, $f(x)$ is the objective function to be minimized, and $g_j(x)$ and $h_k(x)$ represent inequality and equality constraints, respectively.

To handle constraints, we introduce non-negative Lagrange multipliers $\lambda = (\lambda_j)_{j\in\mathcal{J}} \ge 0$ and unconstrained multipliers $\mu = (\mu_k)_{k\in\mathcal{K}} \in \mathbb{R}$, and construct the Lagrangian:

$$L(x, \lambda, \mu) = f(x) + \sum_{j\in\mathcal{J}} \lambda_j g_j(x) + \sum_{k\in\mathcal{K}} \mu_k h_k(x).$$

The dual function is defined as:

$$\theta(\lambda, \mu) = \inf_{x\in\mathcal{X}} L(x, \lambda, \mu).$$

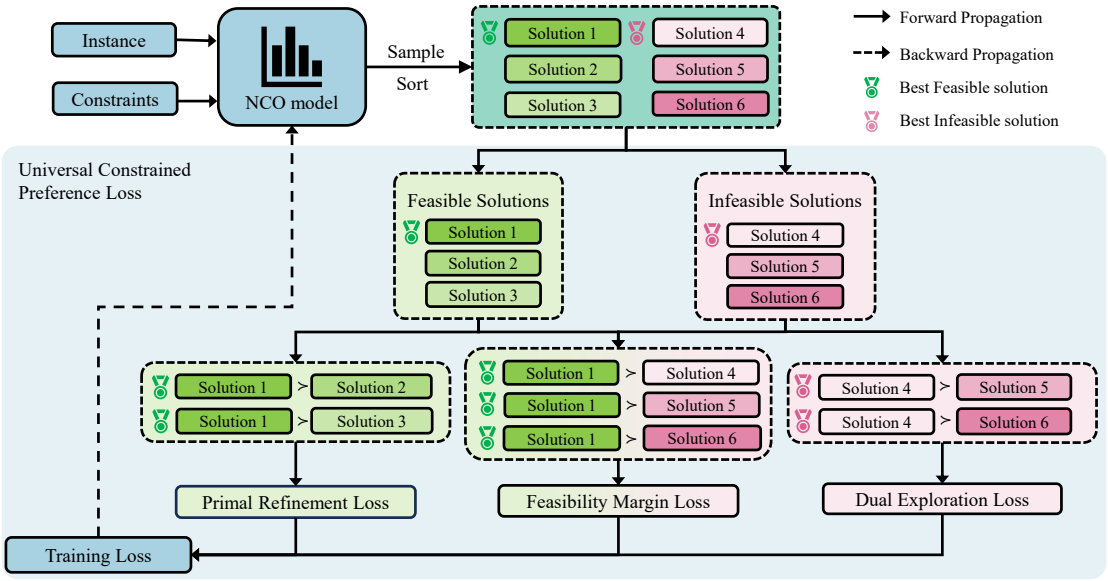


Figure 1: Overview of the Universal Constrained Preference Optimization (UCPO) framework. The process begins by sampling candidate solutions from the base NCO model, followed by fine-tuning the model using Universal Constrained Preference Loss.

To reduce computational complexity, the multipliers are often fixed at certain values $(\bar{\lambda}, \bar{\mu})$, transforming the dual problem into a single-level unconstrained optimization problem:

$$\min_{x \in \mathcal{X}} \left\{ f(x) + \bar{\lambda}^\top g(x) + \bar{\mu}^\top h(x) \right\}.$$

This approach converts the original saddle-point problem over (x, λ, μ) into a single-variable minimization problem over $x \in \mathcal{X}$. As a result, any off-the-shelf unconstrained or model-free algorithm (e.g., REINFORCE (Williams 1992)) can be directly applied without the need for an outer loop to update the multipliers. However, this simplicity comes at the cost of losing strict feasibility guarantees unless $(\bar{\lambda}, \bar{\mu})$ corresponds to an optimal dual solution. In practice, the multipliers are treated as tunable "penalty coefficients", enabling the generation of approximately feasible solutions through manual or heuristic adjustments.

Preference Optimization Methods

Given an instance $x \in \mathcal{X}$ and a policy network with parameters θ, the policy $\pi_\theta(\cdot \mid x)$ generates N candidate solutions:

$$\mathcal{T}_x = \{\tau_i\}_{i=1}^N \sim \pi_\theta(\cdot \mid x).$$

For every ordered pair (τ_i, τ_j), we define the preference label as:

$$y_{ij} = \mathbb{1}\{f(x, \tau_i) < f(x, \tau_j)\},$$

where $f(x, \tau)$ denotes the objective value of solution τ on instance x. The symbol $\mathbb{1}$ indicates that τ_i is strictly better than τ_j.

The Bradley–Terry model (Bradley and Terry 1952) translates the objective gap into a preference probability:

$$p_\theta(\tau_i \succ \tau_j \mid x) = \sigma\big(\beta \cdot d_\theta(x, \tau_i, \tau_j)\big),$$

where $\sigma(\cdot)$ is the logistic (sigmoid) function, $d_\theta(x, \tau_i, \tau_j) = \alpha\big[\log \pi_\theta(\tau_i \mid x) - \log \pi_\theta(\tau_j \mid x)\big]$ represents the log-likelihood difference, $\alpha > 0$ is a temperature parameter, and β is an adaptive scaling factor (BOPO (Liao et al. 2025) sets $\beta = f(x, \tau_j)/f(x, \tau_i)$, while POCO (Pan et al. 2025) simply fixes $\beta = 1$).

Maximizing the likelihood over all preference pairs is equivalent to minimizing the loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{x \sim \mathcal{X}} \frac{1}{|\mathcal{T}_x|^2} \sum_{i \neq j} y_{ij} \log p_\theta(\tau_i \succ \tau_j \mid x).$$

Substituting the Bradley–Terry model yields the closed-form objective:

$$\mathcal{L}(\theta) = -\mathbb{E}_{x \sim \mathcal{X}} \frac{1}{|\mathcal{T}_x|^2} \sum_{i \neq j} y_{ij} \log \sigma\Big(\alpha\beta\big[\log \pi_\theta(\tau_i \mid x) - \log \pi_\theta(\tau_j \mid x)\big]\Big).$$

Methodology

To address complex constrained combinatorial optimization problems, we introduce Universal Constrained Preference Optimization (UCPO), a novel framework that seamlessly integrates preference learning into the combinatorial optimization workflow in a plug-and-play manner. Designed as a lightweight loss function, UCPO can be easily incorporated into any existing RL-based solver without the need for extensive architectural modifications or complete retraining. This is achieved by leveraging a warm-start fine-tuning procedure that utilizes existing pre-trained checkpoints, thereby significantly enhancing performance in constrained settings with minimal additional computational overhead.

The overall architecture of UCPO is illustrated in Figure 1. The detailed components are presented as follows.

Constrained Preference Optimization Formulation

Partial-Order Relation. Consider an instance $x \in \mathcal{X}$ with constraint sets $\mathcal{G} = \{g_j(x) \leq 0, j \in \mathcal{J}\}$ and $\mathcal{H} = \{h_k(x) = 0, k \in \mathcal{K}\}$. The policy $\pi_\theta(\cdot \mid x)$ generates N candidate solutions $\mathcal{T}_x = \{\tau_i\}_{i=1}^N$, some of which may violate the constraints. To address infeasible solutions, we introduce the indicator function:

$$I(x,\tau) = \begin{cases} 1, & \exists\, j \in \mathcal{J} : g_j(x,\tau) > 0 \text{ or} \\ & \exists\, k \in \mathcal{K} : h_k(x,\tau) \neq 0, \\ 0, & \forall\, j \in \mathcal{J} : g_j(x,\tau) \leq 0 \text{ and} \\ & \forall\, k \in \mathcal{K} : h_k(x,\tau) = 0. \end{cases}$$

Here, $I(x,\tau) = 0$ if and only if τ is strictly feasible. Using the quadruple $(\tau_i, \tau_j, I(x,\tau_i), I(x,\tau_j))$, we define the partial order $\tau_i \succ \tau_j$ if exactly one of the following conditions holds:

$$\begin{cases} I(x,\tau_i) = I(x,\tau_j) = 0 \text{ and } f(x,\tau_i) < f(x,\tau_j), \\ I(x,\tau_i) = 0 \text{ and } I(x,\tau_j) = 1, \\ I(x,\tau_i) = I(x,\tau_j) = 1 \text{ and } L(x,\tau_i,\lambda,\mu) < L(x,\tau_j,\lambda,\mu), \end{cases}$$

where

$$L(x,\tau,\lambda,\mu) = f(x,\tau) + \sum_{j\in\mathcal{J}} \lambda_j g_j(x,\tau) + \sum_{k\in\mathcal{K}} \mu_k h_k(x,\tau)$$

is the Lagrangian with non-negative multipliers $\lambda = (\lambda_j)_{j\in\mathcal{J}} \geq 0$ and unconstrained multipliers $\mu = (\mu_k)_{k\in\mathcal{K}} \in \mathbb{R}$. This hierarchy prioritizes: 1) lower objective values among feasible solutions, 2) feasible solutions over infeasible ones, and 3) lower Lagrangian values among infeasible solutions.

Bradley–Terry Model. Following the typical formalization of preference optimization, we convert the partial order into a preference probability. For a solution pair (τ_i, τ_j) and instance x, the preference probability is defined as:

$$p_\theta(\tau_i \succ \tau_j \mid x) = \sigma(\beta\,[\log \pi_\theta(\tau_i \mid x) - \log \pi_\theta(\tau_j \mid x)])\,,$$

where $\log \pi_\theta(\tau \mid x)$ is the log-likelihood of τ under the policy π_θ, $\sigma(\cdot)$ is the logistic sigmoid function, and β is an adaptive scaling factor. Following the parameterization adopted in BOPO (Liao et al. 2025), we dispense with the temperature coefficient α and retain only the scalar β. This formulation efficiently captures the relative preference between solutions while reducing computational complexity.

Design of Universal Constrained Preference Loss

To address environments with complex constraints and sparse feasible solutions, we design three loss components, i.e., $\mathcal{L}^{\text{Dual}}$, $\mathcal{L}^{\text{Margin}}$ and $\mathcal{L}^{\text{Primal}}$, that progressively guide the policy from infeasible exploration to high-quality feasible solution learning. Although not strictly mutually exclusive, these losses follow a natural progression: *Dual Exploration → Feasibility Margin → Primal Refinement*, which enables the model to balance constraint satisfaction with objective optimality in a principled manner.

Given a problem instance and its constraints, the policy network concurrently samples N complete trajectories. While this yields $\binom{N}{2}$ pairwise preferences in theory, we instead enforce the previously established partial order to construct a *single ranked list*:

$$\tau_1 \succ \tau_2 \succ \cdots \succ \tau_N,$$

which immediately partitions the candidates into

$$\mathcal{T}^T = \{\tau \in \mathcal{T}_x \mid I(x,\tau) = 0\} \quad \text{(feasible)},$$
$$\mathcal{T}^F = \{\tau \in \mathcal{T}_x \mid I(x,\tau) = 1\} \quad \text{(infeasible)}.$$

Either subset may be empty, especially during early training stages. Our loss design guides the policy toward strict feasibility while refining objective quality by activating three terms depending on the realization of $\mathcal{T}^T$ and $\mathcal{T}^F$.

Dual Exploration Loss $\mathcal{L}^{\text{Dual}}$. Activated only when $\mathcal{T}^T = \emptyset$, i.e., all sampled trajectories are infeasible. Let $\tau_\circ = \arg\min_{\tau\in\mathcal{T}^F} L(x,\tau,\lambda,\mu)$ be the "least-infeasible" solution under particular Lagrange multiplier. We contrast $\tau_\circ$ against the remaining infeasible trajectories to guide exploration toward lower constraint violation:

$$\mathcal{L}^{\text{Dual}}(\theta) = -\,\mathbb{E}_{x\sim\mathcal{X}} \frac{1}{|\mathcal{T}^F| - 1} \sum_{\tau\in\mathcal{T}^F\setminus\{\tau_\circ\}} \log\sigma\Big(\beta^{\text{Dual}}\big[\log \pi_\theta(\tau_\circ \,|\, x) - \log \pi_\theta(\tau \,|\, x)\big]\Big), \tag{1}$$

with $\beta^{\text{Dual}} = L(x,\tau,\lambda,\mu)/L(x,\tau_\circ,\lambda,\mu) \geq 1$.

It is worth noting that $\mathcal{L}^{\text{Dual}}$ is activated only during the very beginning of training, while no feasible solution has yet been found. Once feasible solutions are sampled (via exploration guided by the Lagrangian-based ranking), the loss is deactivated in favor of $\mathcal{L}^{\text{Margin}}$ and $\mathcal{L}^{\text{Primal}}$, which operate solely within the feasible region and are multiplier-agnostic. Consequently, the long-term policy behavior is dominated by constraint-compliant optimization, decoupling final performance from the initial (λ,μ).

Feasibility Margin Loss $\mathcal{L}^{\text{Margin}}$. Activated when both $\mathcal{T}^T$ and $\mathcal{T}^F$ are non-empty. Let $\tau_\star = \arg\min_{\tau\in\mathcal{T}^T} f(x,\tau)$ be the best feasible solution. We contrast $\tau_\star$ against every infeasible trajectory $\tau \in \mathcal{T}^F$:

$$\mathcal{L}^{\text{Margin}}(\theta) = -\mathbb{E}_{x\sim\mathcal{X}} \frac{1}{|\mathcal{T}^F|} \sum_{\tau\in\mathcal{T}^F} \log\sigma\Big(\beta^{\text{Margin}}\big[\log \pi_\theta(\tau_{r_\star} \,|\, x) - \log \pi_\theta(\tau \,|\, x)\big]\Big), \tag{2}$$

with adaptive scaling $\beta^{\text{Margin}} = L(x,\tau,\lambda,\mu)/f(x,\tau_\star) > 1$. The Lagrangian L smooths the transition into the feasible region, eliminating gradient discontinuities. This loss simultaneously attracts the policy toward the current best feasible solution and repels it from all infeasible ones.

Primal Refinement Loss $\mathcal{L}^{\text{Primal}}$. Activated when $|\mathcal{T}^T| \geq 2$. We refine the policy within the feasible region by pairing $\tau_\star$ with every other feasible solutions $\tau \in \mathcal{T}^T \setminus \{\tau_\star\}$:

$$\mathcal{L}^{\text{Primal}}(\theta) = -\mathbb{E}_{x\sim\mathcal{X}} \frac{1}{|\mathcal{T}^T| - 1} \sum_{\tau\in\mathcal{T}^T\setminus\{\tau_\star\}} \log\sigma\Big(\beta^{\text{Primal}}\big[\log \pi_\theta(\tau_\star \,|\, x) - \log \pi_\theta(\tau \,|\, x)\big]\Big), \tag{3}$$

Algorithm 1: UCPO Training Protocol.

Require: Epochs E_{ft}, batch size B, data distribution S, NCO pretrained checkpoint θ_0.

Initialize $\theta \leftarrow \theta_0$
for epoch $e = 1$ **to** E_{ft} **do**
  for batch $b = 1$ **to** $\lceil E_{\text{ft}}/B \rceil$ **do**
    $\{\langle x_i, \mathcal{G}_i, \mathcal{H}_i \rangle\}_{i=1}^{B} \leftarrow \text{DataGenerator}(S)$
    for each x in mini-batch **do**
      Sample $\tau_{(1\ldots N)} \sim \text{NCO}(\langle x_i, \mathcal{G}_i, \mathcal{H}_i \rangle)$
      Sort and split solutions
        $\mathcal{T}^T \leftarrow \{\tau \in \mathcal{T}_x \mid I(x,\tau) = 0\}$
        $\mathcal{T}^F \leftarrow \{\tau \in \mathcal{T}_x \mid I(x,\tau) = 1\}$
      Compute $\mathcal{L}^{\text{Dual}}(\theta), \mathcal{L}^{\text{Margin}}(\theta), \mathcal{L}^{\text{Primal}}(\theta)$
      $\mathcal{L}(\theta) = \mathcal{L}^{\text{Dual}}(\theta) + \mathcal{L}^{\text{Margin}}(\theta) + \mathcal{L}^{\text{Primal}}(\theta)$
      Update parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
    end for
  end for
end for

scaled by $\beta^{\text{Primal}} = f(x,\tau)/f(x,\tau_\star) \geq 1$. Empirically, we observe that $\beta^{\text{Margin}} > \beta^{\text{Primal}}$ in most instances; the larger gradient on the constraint term implicitly prioritizes feasibility. If $\mid \mathcal{T}^T \mid \leq 1$, we simply set $\mathcal{L}^{\text{Primal}} = 0$.

Overall Training Objective. The policy network is optimized with the composite loss,

$$\mathcal{L}(\theta) = \mathcal{L}^{\text{Dual}}(\theta) + \mathcal{L}^{\text{Margin}}(\theta) + \mathcal{L}^{\text{Primal}}(\theta), \qquad (4)$$

where the active terms are automatically selected from cold start to high performance, ensuring stable end-to-end training across all stages.

Light Warm-Start Fine-Tuning

The training procedure of UCPO, as formalized in Algorithm 1, is designed to seamlessly integrate with existing NCO frameworks. It achieves full architectural compatibility by neither modifying the network topology nor introducing additional learnable parameters. This design allows any pre-trained checkpoint, especially those already fine-tuned for constrained settings, to serve as a high-quality initializer.

Given a pre-trained policy π_{θ_0}, UCPO performs fine-tuning for a limited number of epochs $E_{\text{ft}} \ll E_{\text{base}}$, where E_{base} denotes the original training epochs. The fine-tuning objective minimizes the composite loss defined in Eq. (4) through gradient updates. This process simultaneously enhances constraint satisfaction and solution quality.

The warm-start protocol achieves two key advantages:

- **Few-Overhead Transfer**: By preserving architectural invariance, UCPO ensures that the computational complexity of the pre-trained model remains unchanged. Specifically, the memory footprint and inference latency of the fine-tuned model are equivalent to those of the pre-trained model.
- **Rapid Convergence**: Empirically, UCPO reaches stable performance within $E_{\text{ft}} = \lceil 0.01 E_{\text{base}} \rceil$ to $\lceil 0.05 E_{\text{base}} \rceil$ epochs. This efficiency enables UCPO to achieve significant performance improvements with less than 5% of the original training budget.

Experiments

Experimental Setups

Datasets. Experiments target four canonical constraints: time windows, draft limits, vehicle capacity, and fleet size. Four benchmark tasks are examined accordingly: TSPTW, TSPDL, CVRPTW, and CVRPTWLV. For TSPTW and TSPDL, we adopt the generation protocol proposed by PIP and create instances at three difficulty levels—Easy, Medium, and Hard—based on constraint-satisfaction hardness. CVRPTW and CVRPTWLV follow the setting introduced by JAMPR (Falkner and Schmidt-Thieme 2020) and are generated uniformly at the Hard level. All datasets are fixed at 50 and 100 nodes. Each validation set comprises exactly 10,000 instances.

Baselines. UCPO is evaluated against three representative algorithmic families: 1) Problem-specific heuristics, exemplified by LKH3 (Helsgaun 2017), which are meticulously tailored to each VRP variant. 2) General-purpose RL-NCO, represented by POMO (Kwon et al. 2020) and SLIM (Corsini et al. 2024), that employs reinforcement learning to deliver universal solvers. 3) Constraint-specialized models, including PIP (Bi et al. 2024) and its distilled variant PIP-D.

Benchmark Setup and Base Models. UCPO is warm-started into three backbone models: POMO, PIP, and PIP-D. All hyper-parameters are kept identical to the original publications, and the publicly released best checkpoints are used without modification. Training data distributions, network architectures, and optimizer configurations remain exactly the same. For Lagrangian-based models, the Lagrange multiplier is fixed to 1, consistent with the reported values. During warm-start, POMO and PIP are fine-tuned for up to 500 epochs; PIP-D is trained for 100 epochs. These epoch counts are jointly determined by convergence curves and training-time budgets. Note that the original checkpoints were obtained after 10,000 epochs, so warm-start training amounts to only 1%–5% of the total previous computation. To provide a comprehensive evaluation, UCPO is additionally cold-started on POMO without pre-trained checkpoint. All training and inference are executed on NVIDIA GeForce GTX 1080 Ti GPUs and Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz CPUs.

Evaluation. Inference follows a unified protocol: 1) Sampling-based decoding: the number of samples equals the instance size; 8× data augmentation is enabled. 2) Metrics include the infeasible rate on the validation set, the average objective value of **feasible solutions** and the average optimality gap with respect to LKH3. An instance is regarded as feasible if at least one sampled solution satisfies all constraints; the solution with the smallest objective among the feasible ones is returned. 3) Inference time and memory usage are not reported, as they remain consistent with the backbone models.

Main Results

The performance of UCPO on various benchmark tasks is summarized in Tables 1–3. After fine-tuning with UCPO, the NCO models based on different backbones, including

Method	TSPTW-50									TSPTW-100								
	Easy			Medium			Hard			Easy			Medium			Hard		
	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%
LKH3	0.00	7.31	0.00	0.00	13.02	0.00	0.12	25.61	0.00	0.00	10.21	0.00	0.00	18.74	0.00	0.07	51.24	0.00
POMO	0.00	7.54	3.15	3.77	13.68	5.07	35.25	26.22	2.38	0.00	10.83	6.07	0.12	20.78	10.89	100.00	—	—
UCPO (POMO)	0.00	7.41	1.30	0.08	13.26	1.86	1.32	25.75	0.55	0.00	10.54	3.25	0.01	19.84	5.89	100.00	—	—
UCPO* (POMO)	0.00	**7.36**	**0.64**	0.15	13.19	1.31	0.13	25.61	0.01	0.00	10.58	3.66	0.00	19.93	6.35	**0.87**	51.25	0.02
PIP	0.00	7.50	2.60	0.90	13.40	2.92	2.67	25.66	0.20	0.00	10.57	3.53	0.19	19.61	4.64	16.27	51.42	0.35
UCPO (PIP)	0.00	7.39	1.11	**0.04**	**13.14**	**0.89**	**0.76**	**25.61**	**0.00**	0.00	**10.41**	**1.94**	**0.00**	**19.17**	**2.27**	5.01	51.28	0.07
PIP-D	0.00	7.49	2.46	0.65	13.45	3.30	3.07	25.69	0.31	0.00	10.66	4.41	0.03	19.79	5.60	6.48	51.39	0.29
UCPO (PIP-D)	0.00	7.42	1.49	0.05	13.18	1.23	1.07	25.61	0.00	0.00	10.53	3.11	0.01	19.32	3.08	2.99	**51.24**	**0.00**

Table 1: Performance on TSPTW-50 and TSPTW-100. Bars indicate no feasible solution. UCPO* represents UCPO with cold-start training.

Method	Medium (TSPDL-50)			Hard (TSPDL-50)		
	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%
LKH3	0.00	10.87	0.00	0.00	13.30	0.00
POMO	12.52	10.98	1.01	29.25	13.85	4.10
UCPO (POMO)	4.00	11.04	1.54	0.01	**13.48**	**1.35**
UCPO*(POMO)	0.07	11.09	2.03	**0.00**	13.54	1.83
PIP	0.43	11.22	3.22	2.10	13.66	2.71
UCPO (PIP)	0.02	**11.06**	**1.73**	0.01	13.48	1.36
PIP-D	0.37	11.26	3.59	0.82	13.80	3.76
UCPO (PIP-D)	**0.01**	11.10	2.15	0.04	13.56	1.92
Method	Medium (TSPDL-100)			Hard (TSPDL-100)		
	Inst.%	Obj.	Gap%	Inst.%	Obj.	Gap%
LKH3	0.00	16.39	0.00	0.00	20.70	0.00
POMO	32.16	17.11	4.39	99.85	**20.95**	**1.21**
UCPO (POMO)	**0.02**	17.29	5.49	18.84	21.40	3.38
UCPO*(POMO)	0.03	17.27	5.38	**0.01**	22.37	8.06
PIP	0.38	17.71	8.05	20.66	22.30	7.73
UCPO (PIP)	**0.02**	**17.19**	**4.86**	0.04	22.40	8.22
PIP-D	0.23	17.84	8.85	7.91	22.84	10.34
UCPO (PIP-D)	**0.02**	17.44	6.42	**0.01**	22.42	8.30

Table 2: Performance on TSPDL-50 and TSPDL-100.

Method	CVRPTW50		CVRPTWLV50	
	Inst. %	Obj.	Inst. %	Obj.
POMO	0.20 %	14.13	3.26 %	14.20
UCPO (POMO)	**0.09** %	13.58	0.98 %	13.61
UCPO* (POMO)	0.75 %	**13.50**	2.20 %	**13.56**
SLIM	0.15 %	14.10	2.74 %	14.15
UCPO (SLIM)	0.11 %	13.51	**0.85** %	13.59

Table 3: Performance on CVRPTW50 and CVRPTWLV50.

Method	Medium		Hard	
	Inst. %	Gap %	Inst. %	Gap %
LKH3	0.00 %	0.00 %	0.12 %	0.00 %
UCPO-PIP (λ=0.5)	**0.01** %	0.95 %	**0.24** %	0.00 %
UCPO-PIP (λ=1)	0.04 %	**0.89** %	0.76 %	0.00 %
UCPO-PIP (λ=2)	0.03 %	0.94 %	0.78 %	**-0.01** %
UCPO-POMO (λ=0.5)	**0.03** %	1.97 %	1.42 %	0.61 %
UCPO-POMO (λ=1)	0.08 %	**1.86** %	**1.32** %	**0.55** %
UCPO-POMO (λ=2)	0.09 %	1.87 %	1.77 %	0.61 %

Table 4: Ablation on TSPTW-50 with different λ values.

POMO, PIP, and PIP-D, consistently achieve substantial improvements in both feasibility rate and objective value across diverse test sets.

As illustrated in Table 1, UCPO achieves remarkable performance on TSPTW instances. First, it significantly reduces infeasibility rates. On the Easy and Medium sets, UCPO attains near-zero infeasibility rates (all below 0.15%), significantly outperforming PIP (0.90%) and PIP-D (0.65%). On the Hard set, UCPO* (POMO) reduces POMO's infeasibility rate from 100% to 0.87%. Second, UCPO substantially improves solution quality. On TSPTW-50-Easy, UCPO* (POMO) reduces the optimality gap from 3.15% to 0.64%, achieving superior performance compared to all other methods. On TSPTW-100-Easy, UCPO (PIP) lowers the gap to 1.94%, outperforming PIP's 3.53%. Third, UCPO exhibits strong generalization and convergence across different backbones. It consistently enhances performance across various policy networks (POMO, PIP, PIP-D), reducing the performance gap between baseline models. For example, UCPO (PIP) ranks first on four out of six subsets. Fourth, UCPO* achieves substantial infeasibility reduction (e.g., from 16.27% to 0.87% on TSPTW-100-Hard), although it requires significantly more training time to converge.

Tables 2 and 3 further demonstrate UCPO's consistent improvements in feasibility and objective values over baseline methods. On TSPDL-100-Hard, UCPO* (POMO) reduces infeasibility from 99.85% to 0.01%, achieving better trade-offs. Even in cold-start mode on CVRPTW, UCPO* achieves the best objective values (13.50 vs. 14.13), show-

Loss			Medium		Hard	
Feasibility	Primal	Dual	Inst. %	Gap %	Inst. %	Gap %
✓			0.29 %	1.87 %	1.74 %	0.64 %
	✓		0.30 %	**1.84** %	1.91 %	0.62 %
		✓	100.00 %	— %	100.00 %	— %
✓	✓		0.21 %	1.87 %	1.94 %	0.59 %
✓		✓	0.14 %	1.90 %	1.53 %	0.62 %
	✓	✓	0.19 %	1.90 %	2.19 %	0.62 %
✓	✓	✓	**0.08** %	1.86 %	**1.32** %	**0.55** %
✓	✓		100.00 %	— %	100.00 %	— %
✓	✓	✓	**0.08** %	**1.31** %	**1.61** %	**0.01** %

Table 5: Ablation of loss terms on TSPTW-50. The upper pane presents results obtained with warm-started models; the lower pane corresponds to models trained from scratch.

casing its strong generalization capability under hard constraints.

Sensitivity Analysis

To verify the robustness of UCPO with respect to the Lagrange multiplier λ, we conduct experiments on 50-node TSPTW instances on both Medium and Hard levels. Using POMO and PIP as base models, we train and evaluate with $\lambda \in \{0.5, 1, 2\}$.

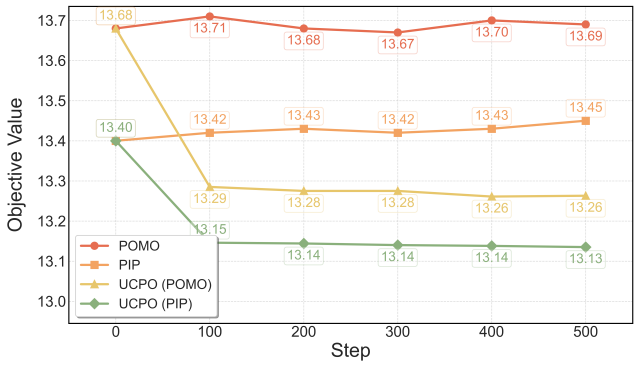
As shown in Table 4, the maximal deviations in infeasibility rate and objective gap across different λ are merely 0.54% and 0.11%, respectively. Moreover, neither metric exhibits a monotonic trend with respect to λ. The observed variations can be attributable to training stochasticity. These results indicate that UCPO exhibits minimal sensitivity to its sole hyperparameter λ, underscoring its exceptional parameter robustness. This characteristic is particularly significant in practical applications, as it eliminates the need for meticulous hyperparameter tuning and enhances the general applicability of UCPO across diverse constrained optimization scenarios.

We further train and evaluate with four random seeds $\{2023, 2024, 2025, 2026\}$, achieving standard deviations below 0.01%, consistently outperforming existing methods.

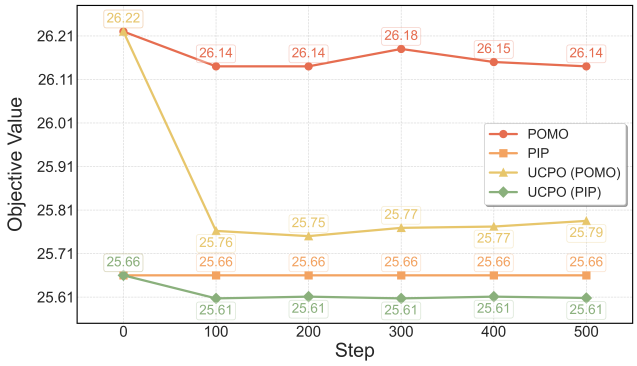
Ablation Studies

We conduct systematic ablations to assess the contributions of the three components of our universal constrained loss function and the impact of warm-start training epochs.

As shown in Table 5, experiments on 50-node TSPTW instances (Medium and Hard difficulty levels) yield two key insights. 1) Both the *Feasibility Margin Loss* and *Primal Refinement Loss* are essential for achieving optimal performance. Removing either component results in a measurable decline in feasibility or solution quality, whereas combining these two losses consistently delivers superior outcomes. 2) The *Dual Exploration Loss*, although inactive during warm-start training (due to the high feasibility of generated trajectories), proves indispensable in cold-start scenarios, enabling rapid convergence to feasible solutions.



(a) Medium



(b) Hard

Figure 2: Post-training values across two difficulty levels (Medium, Hard).

Figure 2 further illustrates the efficiency of our warm-start approach. After further training from converged POMO or PIP checkpoints yields minimal improvements, UCPO achieves significant performance gains after just 100 warm-start steps, representing merely 1% of the original training budget.

Conclusion

We introduce Universal Constrained Preference Optimization (UCPO), a novel framework that addresses the feasibility–performance trade-off in neural combinatorial optimization (NCO) with complex constraints. UCPO is designed as a plug-and-play module that can be integrated with any existing NCO model without requiring architectural changes or manual tuning of Lagrange multipliers. By leveraging a warm-start fine-tuning procedure, UCPO achieves near-100% feasibility and near-optimal solutions on complex constrained tasks with just 1%–5% of the original training budget.

For future research, we propose three directions: 1) developing mechanisms for UCPO to handle dynamic constraints in real-time applications, 2) extending UCPO with meta-learning capabilities to enable rapid adaptation to unseen constraints, and 3) decoupling the constraint module to allow a single base model to generalize across diverse constraint settings through module swapping.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62472461), the Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515010129, 2024A1515010871).

References

Bi, J.; Ma, Y.; Zhou, J.; Song, W.; Cao, Z.; Wu, Y.; and Zhang, J. 2024. Learning to Handle Complex Constraints for Vehicle Routing Problems. In *Advances in Neural Information Processing Systems*.

Bradley, R. A.; and Terry, M. E. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4): 324–345.

Chen, J.; Gong, Z.; Liu, M.; Wang, J.; Yu, Y.; and Zhang, W. 2024. Looking ahead to avoid being late: Solving hard-constrained traveling salesman problem. *arXiv preprint arXiv:2403.05318*.

Chen, J.; Huang, H.; Zhang, Z.; and Wang, J. 2022. Deep reinforcement learning with two-stage training strategy for practical electric vehicle routing problem with time windows. In *International Conference on Parallel Problem Solving from Nature*.

Corsini, A.; Porrello, A.; Calderara, S.; and Dell'Amico, M. 2024. Self-labeling the job shop scheduling problem. *Advances in Neural Information Processing Systems*.

Drakulic, D.; Michel, S.; and Andreoli, J.-M. 2025. GOAL: A Generalist Combinatorial Optimization Agent Learner. *International Conference on Learning Representations*.

Fadda, P.; Mancini, S.; Serra, P.; and Fancello, G. 2023. The Heterogeneous Fleet Vehicle Routing Problem with Draft Limits. *Computers Operations Research*, 149: 106024.

Falkner, J. K.; and Schmidt-Thieme, L. 2020. Learning to solve vehicle routing problems with time windows through joint attention. *arXiv preprint arXiv:2006.09100*.

Fang, H.; Song, Z.; Weng, P.; and Ban, Y. 2024. INViT: A Generalizable Routing Problem Solver with Invariant Nested View Transformer. In *Proceedings of the 41st International Conference on Machine Learning*.

Gao, C.; Shang, H.; Xue, K.; Li, D.; and Qian, C. 2024. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *Proceedings of the 43rd International Joint Conference on Artificial Intelligence*.

Helsgaun, K. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12: 966–980.

Hottung, A.; Mahajan, M.; and Tierney, K. 2025. PolyNet: Learning Diverse Solution Strategies for Neural Combinatorial Optimization. In *The 13th International Conference on Learning Representations*.

Kim, M.; Park, J.; and Park, J. 2022. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *International Conference on Learning Representations*.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*.

Kwon, Y.-D.; Choo, J.; Yoon, I.; Park, M.; Park, D.; and Gwon, Y. 2021. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*.

Liao, Z.; Chen, J.; Wang, D.; Zhang, Z.; and Wang, J. 2025. BOPO: Neural Combinatorial Optimization via Best-anchored and Objective-guided Preference Optimization. In *42nd International Conference on Machine Learning*.

Liu, F.; Lin, X.; Wang, Z.; Zhang, Q.; Xialiang, T.; and Yuan, M. 2024. Multi-task learning for routing problem with cross-problem zero-shot generalization. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1898–1908.

Liu, X.; Chen, Y.-L.; Por, L. Y.; and Ku, C. S. 2023. A Systematic Literature Review of Vehicle Routing Problems with Time Windows. *Sustainability*, 15(15).

Luo, F.; Lin, X.; Liu, F.; Zhang, Q.; and Wang, Z. 2023. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*.

Ma, Y.; Cao, Z.; and Chee, Y. M. 2023. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. *Advances in Neural Information Processing Systems*.

Ma, Y.; Li, J.; Cao, Z.; Song, W.; Zhang, L.; Chen, Z.; and Tang, J. 2021. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*.

Meng, Y.; Xia, M.; and Chen, D. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*.

Pan, M.; Lin, G.; Luo, Y.-W.; Zhu, B.; Dai, Z.; Sun, L.; and Yuan, C. 2025. Preference Optimization for Combinatorial Optimization Problems. In *42nd International Conference on Machine Learning*.

Pan, X.; Jin, Y.; Ding, Y.; Feng, M.; Zhao, L.; Song, L.; and Bian, J. 2023. H-tsp: Hierarchically solving the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sun, Z.; and Yang, Y. 2023. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in Neural Information Processing Systems*.

Tang, Q.; Kong, Y.; Pan, L.; and Lee, C. 2022. Learning to solve soft-constrained vehicle routing problems with lagrangian relaxation. *arXiv preprint arXiv:2207.09860*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *Advances in Neural Information Processing Systems*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3): 229–256.

Xia, Y.; and Zhang, X. 2024. A Neural Column Generation Approach to the Vehicle Routing Problem with Two-Dimensional Loading and Last-In-First-Out Constraints. In *International Joint Conference on Artificial Intelligence*.

Ye, H.; Wang, J.; Cao, Z.; Liang, H.; and Li, Y. 2023. DeepACO: Neural-enhanced ant systems for combinatorial optimization. *Advances in Neural Information Processing Systems*.

Zhou, J.; Cao, Z.; Wu, Y.; Song, W.; Ma, Y.; Zhang, J.; and Xu, C. 2024. MVMoE: Multi-Task Vehicle Routing Solver with Mixture-of-Experts. In *International Conference on Machine Learning*.