# SMOP: SELF-SUPERVISED MULTI-PERIOD OPTIMIZATION PROXY FOR RESOURCE ALLOCATION WITH TEMPORAL CONSTRAINTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Multi-period resource allocation problems are typically solved using optimization tools, but solving optimization from scratch can be computationally expensive for large-scale online decision-making. Recent neural-network-based optimization proxies offer substantial speedups, yet they often struggle to preserve feasibility over an entire horizon when global resource-balance constraints interact with inter-temporal constraints: a correction made at the current period may destroy future feasibility. To address this challenge, we exploit the underlying resource–temporal structure and develop a Self-supervised Multi-period Optimization Proxy (SMOP) with three main components: backward–forward feasible-interval propagation, neural-network-based proportional prediction with projection-based feasibility repair, and future-aware forward recovery. We establish full-horizon feasibility guarantees while reducing the nominal computational complexity from cubic to linear in problem size compared with the dense KKT-based differentiable optimization. Experiments on fixed commitment economic dispatch demonstrate that the proposed method achieves millisecond-level inference and produces solutions with zero constraint violations. Across most test scenarios, the resulting feasible solutions also exhibit less than 0.5% optimality gaps without using labels, demonstrating that SMOP can achieve both high computational efficiency and strong solution quality.

## 1 INTRODUCTION

Multi-period resource allocation is fundamental to decision-making in applications such as energy dispatch, dynamic traffic management, supply-chain scheduling, and industrial production planning. These problems coordinate multiple resources over time while satisfying system-wide requirements, resource-specific operating limits, and intertemporal constraints (Boyd et al., 2017). Conventional approaches repeatedly solve the underlying optimization problem using numerical solvers (Wu et al., 2026). Although reliable, this process can become a computational bottleneck in large-scale or latency-sensitive applications that require decisions for many problem instances.

Machine-learning-based optimization proxies have recently emerged as a promising alternative by learning a direct mapping from problem parameters to decision variables (Chen et al., 2023; Park & Van Hentenryck, 2024; Nguyen & Donti, 2025; Donti et al., 2021). Early approaches primarily relied on supervised learning using optimal solutions generated by conventional solvers (Chen et al., 2022). Producing such labels, however, can be computationally expensive, while the resulting predictions may still violate constraints, particularly under distribution shifts. Self-supervised optimization proxies[1] instead learn directly from the optimization model, thereby avoiding expensive optimal labels (Park & Van Hentenryck, 2023; Chen et al., 2023). Existing approaches mainly enforce feasibility through either penalty terms in the training loss or hard feasibility layers, such as projection, completion, or repair layers (Donti et al., 2021; Chen et al., 2023). Penalty-based methods treat constraints softly and therefore cannot guarantee feasibility, while balancing multiple penalties may complicate training and degrade solution quality. Hard feasibility layers, such as projection, completion, or repair layers, provide stronger constraint guarantees, but existing methods largely focus on single-period constraints (Chen et al., 2023; Park & Van Hentenryck, 2024; Nguyen & Donti, 2025; Donti et al., 2021). For multi-period problems with cross-resource and intertemporal coupling, enforcing feasibility often requires solving a coupled differentiable projection problem (Amos & Kolter, 2017; Agrawal et al., 2019), which can limit scalability. We refer to the simultaneous satisfaction of period-wise and intertemporal constraints over the entire scheduling horizon as **full-horizon feasibility**. SMOP instead exploits problem structure by constructing future-aware feasible

[1] A self-supervised optimization proxy trains a neural network directly from the original optimization objective and constraints, without requiring optimal-solution labels.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

intervals through backward–forward propagation and applying sequential projection-based repair, thereby preserving full-horizon feasibility under the conditions established in Section 4 while maintaining high computational efficiency.

The feasible set of a multi-period resource-allocation problem exhibits two complementary forms of structure: cross-resource coupling within each period and temporal coupling along the trajectory of each resource. The former arises from system-level requirements, such as demand balance and reserve adequacy, whereas the latter arises from resource dynamics, such as ramping and transition limits. We exploit this structure to develop a self-supervised optimization proxy that coordinates backward–forward interval tightening, exact bounded balance repair through projection, and ramp-aware forward recovery. The proposed framework is developed for a family of multi-period resource-allocation linear programs and instantiated using the fixed-commitment economic dispatch problem.

Our main contributions are summarized as follows:

- **Self-supervised multi-period optimization proxy.** We formulate a class of multi-period resource-allocation linear programs with cross-resource and temporal coupling and propose SMOP, a label-free optimization proxy that preserves full-horizon feasibility through three structure-aware components: backward–forward interval propagation, exact bounded balance projection, and ramp-aware forward recovery.
- **Full-horizon feasibility and linear computational complexity.** We establish explicit conditions for full-horizon feasibility and show that SMOP achieves $\mathcal{O}(TG)$ complexity with a fixed number of projection iterations, avoiding the substantially higher cost of dense KKT-based differentiable optimization methods.
- **Baseline comparison and comprehensive ablation study.** We evaluate SMOP on three fixed-commitment economic dispatch benchmarks and show that it achieves stronger feasibility, smaller optimality gaps, and millisecond-level inference with substantial speedups over conventional solvers and representative learning-based baselines. We further conduct comprehensive ablation and sensitivity studies on optimality-gap convergence, component configurations, numerical precision, projection iterations, and reserve feasibility.

## 2 Problem Formulation

We consider the multi-period, multi-resource linear program with temporal dynamics and system-level coupling constraints. Many applications fall within this family of operations research problems, including multi-plant production planning, supply-chain scheduling, dynamic traffic flow, and fixed-commitment economic dispatch. We use fixed-commitment economic dispatch as the principal illustrative application.

**Multi-Period Resource-Allocation Problem.** We consider a multi-period resource-allocation problem with a set of resources $g \in \mathcal{G} = \{1, \dots, G\}$ over a planning horizon $\mathcal{T} = \{1, \dots, T\}$. Let $\boldsymbol{\xi}$ denote the parameters of a problem instance. For each resource $g$ and period $t$, $p_{g,t}$ denotes the primary allocation and $r_{g,t}$ denotes the available backup or reserve contribution. The problem is formulated as

$$\min_{\boldsymbol{p},\boldsymbol{r}} \quad \sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} \phi_{g,t}(p_{g,t};\boldsymbol{\xi}) \tag{1}$$

$$\text{s.t.} \quad \sum_{g\in\mathcal{G}} p_{g,t} = D_t, \qquad \forall t \in \mathcal{T}, \tag{2}$$

$$\sum_{g\in\mathcal{G}} r_{g,t} \geq R_t, \qquad \forall t \in \mathcal{T}, \tag{3}$$

$$L_{g,t}(p_{g,t-1};\boldsymbol{\xi}) \leq p_{g,t} \leq U_{g,t}(p_{g,t-1};\boldsymbol{\xi}), \qquad \forall g \in \mathcal{G},\ t \in \mathcal{T}, \tag{4}$$

$$0 \leq r_{g,t} \leq P^{\max}_{g,t}(\boldsymbol{\xi}) - p_{g,t}, \qquad \forall g \in \mathcal{G},\ t \in \mathcal{T}. \tag{5}$$

Here, $D_t$ and $R_t$ denote the aggregate demand and required reserve capacity at period $t$, respectively. The functions $L_{g,t}$ and $U_{g,t}$ characterize the dynamically reachable lower and upper bounds of resource $g$, which depend on the previous allocation $p_{g,t-1}$ and the instance parameters $\boldsymbol{\xi}$. The initial allocation $p_{g,0}$ is assumed to be given. The quantity $P^{\max}_{g,t}(\boldsymbol{\xi}) - p_{g,t}$ represents the remaining capacity available for reserve, while $\phi_{g,t}$ denotes a linear or convex piecewise-linear operating-cost function.

We use fixed-commitment economic dispatch as a representative application of equation 1 (Frangioni & Gentile, 2006; Bacci et al., 2021). In this setting, the operating cost can be written as $\phi_{g,t}(p_{g,t};\boldsymbol{\xi}) = c_{g,t}p_{g,t}$, and the reachable dispatch bounds are $L_{g,t}(p_{g,t-1};\boldsymbol{\xi}) = \max\left\{P^{\min}_g u_{g,t},\ p_{g,t-1} - \Psi_{g,t}\right\}$, and $U_{g,t}(p_{g,t-1};\boldsymbol{\xi}) = \min\left\{P^{\max}_g u_{g,t},\ p_{g,t-1} + \Phi_{g,t}\right\}$, where $\Phi_{g,t} = RU_g u_{g,t-1} + SU_g v_{g,t}, \Psi_{g,t} = RD_g u_{g,t} + SD_g w_{g,t}$. Here,

$u_{g,t}, v_{g,t}, w_{g,t} \in \{0,1\}$ are the known commitment, startup, and shutdown decisions, while $RU_g, SU_g, RD_g$, and $SD_g$ denote the corresponding ramping parameters. Accordingly, the instance parameters can be summarized as $\boldsymbol{\xi} = \{P_g^{\max}, P_g^{\min}, \Phi_{g,t}, \Psi_{g,t}, u_{g,t}\}$, and the reserve-capable headroom in equation 5 becomes $P_{g,t}^{\max}(\boldsymbol{\xi}) - p_{g,t} = P_g^{\max} u_{g,t} - p_{g,t}$.

## 3 SMOP: Self-supervised Multi-period Optimization Proxy

We aim to efficiently predict dispatch $p_{g,t}$ and reserve $r_{g,t}$ that strictly satisfy the demand-balance constraint in equation 2, generator operating limits and ramping constraints in equation 4, reserve requirements in equation 3 and equation 5, while maintaining a small optimality gap.

To achieve this, SMOP exploits the interactions among these constraints through three core components. Before training, a *backward–forward interval bounds propagation* procedure computes future-aware feasible intervals for $p_{g,t}$ by jointly accounting for generator limits, full-horizon ramping constraints, and demand feasibility. During training, a multi-period prediction–repair mechanism generates feasible dispatch trajectories: the neural network predicts unit-wise dispatch proportions within the tightened intervals for the current period, and a sequential projection repair enforces demand balance while dynamically updating future intervals to preserve ramping feasibility. After training, reserve is computed analytically from the remaining generation headroom using a proportional allocation rule. Since reserve has no direct contribution to the objective and is fully determined by residual capacity, it is excluded from the learning process. As established in Section 4, the resulting analytic allocation guarantees reserve feasibility. The detailed algorithms are presented in Appendix A.

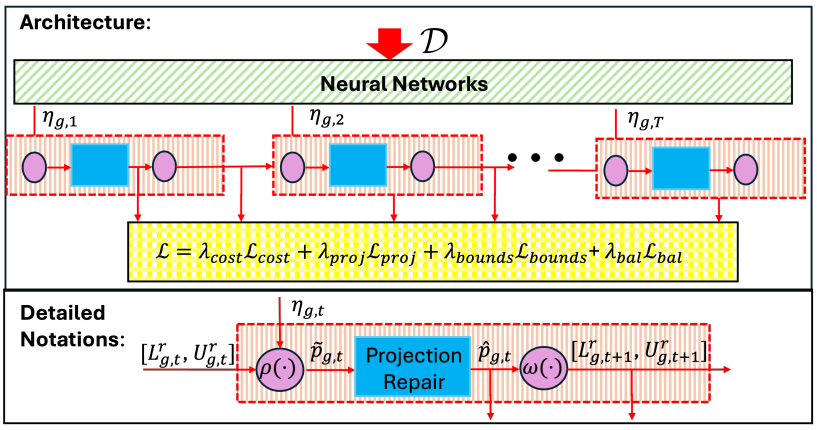


Figure 1: Our Self-supervised Multi-period Optimization Proxy (SMOP) first predicts $\tilde{p}_{g,t}$ within the feasible interval by $\rho(\cdot)$ : $\tilde{p}_{g,t} = L^r_{g,t} + \eta_{g,t}(U^r_{g,t} - L^r_{g,t})$, and then applies a period-by-period *projection repair* to enforce demand balance. After obtaining $\hat{p}_{g,t}$, the next-period feasible interval $[L^r_{g,t+1}, U^r_{g,t+1}]$ is dynamically updated by $\omega(\cdot)$, the *sequential forward recovery,* to preserve ramping feasibility.

### 3.1 Pre-train Forward-Backward Interval Bounds Propagation

Our approach is motivated by the physical interpretation of the ramping constraints in equation 4. Viewed backward from period $t+1$, the ramp-up constraint imposes a lower bound on the current output $p_{g,t}$: the unit must generate sufficiently at period $t$ to reach $p_{g,t+1}$ within its ramp-up capability. Conversely, the ramp-down constraint imposes an upper bound on $p_{g,t}$, preventing the current output from being too high to decrease to $p_{g,t+1}$ within the ramp-down limit. Viewed forward from period $t-1$, the ramp-down constraint determines a lower reachable bound on $p_{g,t}$, while the ramp-up constraint determines its upper reachable bound. Motivated by these bidirectional relationships, we construct backward–forward reachable intervals and further tighten them using the demand-balance constraint in equation 2, thereby incorporating both temporal and system-wide feasibility information. Check Algorithm 2 for details.

**Backward-forward Interval Bound Propagation.** The backward-forward pass computes dispatch intervals that are consistent with past generator limits and ramping constraints. Suppose that, at time $t$, the dispatch of generator $g$ must satisfy $p_{g,t} \in [L^B_{g,t}, U^B_{g,t}]$, where $L^B_{g,t}$ and $U^B_{g,t}$ denote the lower and upper feasible bounds, respectively. Starting from the initial condition $L^B_{g,0} = U^B_{g,0} = p_{g,0}$, the forward propagation uses the ramping constraints to obtain $p_{g,t} \geq L^B_{g,t-1} - \Psi_{g,t}, p_{g,t} \leq U^B_{g,t-1} + \Phi_{g,t}$, which are then intersected with the generation limits $P_g^{\min} u_{g,t} \leq p_{g,t} \leq P_g^{\max} u_{g,t}$. Accordingly, the forward-tightened bounds are $L^B_{g,t} = \max\{P_g^{\min} u_{g,t},\ L^B_{g,t-1} - \Psi_{g,t}\}$, and

$U^B_{g,t} = \min\left\{P^{\max}_g u_{g,t},\ U^B_{g,t-1} + \Phi_{g,t}\right\}$. The backward pass further tightens these intervals using future feasibility. From the ramping relations between periods $t$ and $t+1$, the current dispatch must satisfy $p_{g,t} \geq L^B_{g,t+1} - \Phi_{g,t+1}, p_{g,t} \leq U^B_{g,t+1} + \Psi_{g,t+1}$. Combining these conditions with the generation limits gives the backward-tightened bounds $L^B_{g,t} = \max\left\{P^{\min}_g u_{g,t},\ L^B_{g,t+1} - \Phi_{g,t+1}\right\}$, and $U^B_{g,t} = \min\left\{P^{\max}_g u_{g,t},\ U^B_{g,t+1} + \Psi_{g,t+1}\right\}$. We repeatedly perform forward propagation for $t = 1, \ldots, T$ and backward propagation for $t = T-1, \ldots, 1$ until the bounds converge. The resulting interval $[L^B_{g,t}, U^B_{g,t}]$ characterizes a dispatch range that is consistent with generator limits and ramping feasibility over the full horizon.

**Demand-Aware Bound Tightening.** After computing backward reachable intervals $p_{g,t} \in \left[L^B_{g,t}, U^B_{g,t}\right]$ we can further tighten these intervals using the system load-aware constraint. The purpose is to make the generator-wise intervals not only ramp-aware, but also demand-aware. At each time period $t$, the total dispatch needs to satisfy the load-balance constraint in equation 2. Suppose every generator $j \neq g$ must satisfy $p_{j,t} \in [L^B_{j,t}, U^B_{j,t}]$. Then generator $g$'s output cannot be arbitrary within its own interval. Meanwhile, its output must also satisfy load balance in equation 2. Therefore, we can tighten the backward interval as $L^B_{g,t} = \max\{L^B_{g,t},\ D_t - \sum_{j\neq g} U^B_{j,t}\}$, and $U^B_{g,t} = \min\{U^B_{g,t},\ D_t - \sum_{j\neq g} L^B_{j,t}\}$. This further tightening removes dispatch values that are individually feasible for generator $g$, but impossible to combine with the other generators while satisfying load balance.

### 3.2 Multi-period Prediction–Repair

When ramping constraints are present, the neural-network predictions $\tilde{p}_{g,t}$ across time are no longer independent because each dispatch decision affects the feasible range of subsequent periods. We therefore adopt a sequential forward prediction scheme from $t = 1$ to $T$. At each period, the network predicts $\tilde{p}_{g,t}$ within $[L^r_{g,t}, U^r_{g,t}]$, which is initialized by the pre-train bounds $L^B_{g,t}, U^B_{g,t}$. Then they passed through a projection-based repair layer, as illustrated in Figure 1, to enforce feasibility before proceeding to the next period.

**Exact Bounded Balance Projection Repair.** For each period $t$, consider the projection $\widehat{p}_{g,t} \in \arg\min_{p_{g,t}\in\mathcal{F}} \frac{1}{2}\sum_{g=1}^{G}\left(p_{g,t} - \widetilde{p}_{g,t}\right)^2$, where the feasible region $\mathcal{F} = \{\sum_{g=1}^{G} p_{g,t} = D_t, L^r_{g,t} \leq p_{g,t} \leq U^r_{g,t}, g = 1, \ldots, G.\}$. The solution has the water-filling form (Boyd & Vandenberghe, 2004) $\widehat{p}_{g,t} = \mathrm{clip}(\widetilde{p}_{g,t} - \lambda_t, L^r_{g,t}, U^r_{g,t})$ where the scalar $\lambda_t$ is chosen through bisection (Kiwiel, 2008) method such that $\sum_{g=1}^{G} \mathrm{clip}\left(\widetilde{p}_{g,t} - \lambda_t, L^r_{g,t}, U^r_{g,t}\right) = D_t$. The algorithm details are revealed in Algorithm 3.

**Sequential Forward Recovery.** Given the recovered dispatch $\hat{p}_{g,t-1}$ at time $t-1$, SMOP exploits it to obtain the forward ramp-feasible lower bound $L^r_{g,t} = \max\left\{P^{\min}_g u_{g,t}, \widehat{p}_{g,t-1} - \Psi_{g,t}\right\}$, and the forward ramp-feasible upper bound $U^r_{g,t} = \min\left\{P^{\max}_g u_{g,t}, \widehat{p}_{g,t-1} + \Phi_{g,t}\right\}$. The forward intervals are intersected with the backward-forward interval propagation: $L^r_{g,t} = \max\left\{L^r_{g,t}, L^B_{g,t}\right\}, U_{g,t} = \min\left\{U^r_{g,t}, U^B_{g,t}\right\}$. The demand-aware tightening is then applied. We describe the specific procedures in Algorithm 4.

**Neural Prediction.** We require the dispatch to fall into the feasible range at the ratio $\eta_{g,t}$, i.e., $\tilde{p}_{g,t} = L^r_{g,t} + \eta_{g,t}\left(U^r_{g,t} - L^r_{g,t}\right)$, where $\eta_{g,t} = f_\theta(\mathcal{D})$ is the output of neural networks satisfying $\eta_{g,t} \geq 0, \sum_g \eta_{g,t} = 1$, and the neural-network input $\mathcal{D}^N = \{(\boldsymbol{u}^k, \boldsymbol{v}^k, \boldsymbol{w}^k, \boldsymbol{D}^k, \boldsymbol{R}^k), P^{\min}, P^{\max}, RU, RD, SU, SD, \boldsymbol{p}_0, \boldsymbol{u}_0, \boldsymbol{c}\}_{k=1}^{N}$ and $\mathbf{c}$ represents the generation-cost parameters. Our model employs a $L$-layer bidirectional gated recurrent unit (BiGRU) network to map a $T$-period unit-commitment instance to generator allocation ratios $\eta_{g,t}$ for all $g, t$. At each period, the input consists of generator-level feature groups and available capacity ranges, together with system-level demand and reserve requirements. After layer normalization, the BiGRU produces a temporal representation that captures both past and future operating conditions. A generator-wise feedforward head then maps each temporal representation to $G$ allocation ratios through a softmax activation. These ratios are used to construct the preliminary dispatch $\tilde{p}_{g,t}$, which is subsequently passed to the Exact Bounded Balance Projection Repair layer to obtain the nearest dispatch satisfying demand balance while remaining within the prescribed feasible bounds. If $\sum_g L^r_{g,t} \leq D_t \leq \sum_g U^r_{g,t}$, then this construction satisfies load balance: $\sum_g \tilde{p}_{g,t} = D_t$, and keeps the dispatch within the tightened interval: $L^r_{g,t} \leq \tilde{p}_{g,t} \leq U^r_{g,t}$.

**Correcting dispatch infeasibility through learning.** Since the demand $D_t$ may fall outside the reachable interval, especially during the early stage of training, we define the current-period capacity shortage as $s^{\mathrm{short}}_t = [D_t - \sum_g U^r_{g,t}]_+$ and the minimum-generation excess as $s^{\mathrm{spill}}_t = [\sum_g L^r_{g,t} - D_t]_+$. The maximum violation is treated as the bound-feasibility loss, $\mathcal{L}_{\mathrm{bounds}} = \| \max_t \frac{s^{\mathrm{short}}_t + s^{\mathrm{spill}}_t}{\max(|D_t|, \varepsilon)} \|_2^2$, where $\varepsilon > 0$. This loss not only measures

whether the current demand lies outside the reachable interval, but also provides a corrective learning signal to earlier dispatch decisions that caused the interval infeasibility. Proposition 4 provides the explanations on the corrective gradient to reduce bound infeasibility.

**Proportional Reserve Allocation.** After obtaining the feasible dispatch $\widehat{p}_{g,t}$, SMOP allocates reserve proportionally to the available headroom as $r_{g,t} = R_t \frac{h_{g,t}}{\sum_{j=1}^{G} h_{j,t}}$, where available headroom is $h_{g,t} = P_g^{\max} u_{g,t} - \widehat{p}_{g,t}$. Sufficient headroom is guaranteed by the feasible commitment condition, $D_t + R_t \leq \sum_g P_g^{\max} u_{g,t}$. Under this condition, Proposition 2 guarantees that the proportional reserve allocation satisfies both the system reserve requirement and the individual reserve-capacity limits.

### 3.3 Training Loss

**Generation cost.** The primary training objective is the generation cost, while auxiliary loss terms guide the neural network toward feasible predictions and reduce the correction required by the projection layer. The generation-cost loss of one sample is defined as $\mathcal{L}_{\text{cost}} = \dfrac{1}{S_{\text{scale}} N} \sum_{k=1}^{N} \sum_{g=1}^{G} \sum_{t=1}^{T} c_{g,t} \widehat{p}_{k,g,t}$, where $\widehat{p}_{k,g,t}$ is the $k$th repaired feasible dispatch. In practice, the scale base $S_{\text{scale}} = \max(C_{\min}, \varepsilon)$, where $C_{\min}$ is the given minimum power cost of $G$ generators in $T$ period, and $\varepsilon$ is a small positive constant.

**Projection-alignment loss.** To encourage the neural prediction to approach the feasible solution before repair, we introduce a projection-alignment loss $\mathcal{L}_{\text{proj}} = \frac{1}{NGT} \sum_{g=1}^{G} \sum_{t=1}^{T} \| \frac{\widetilde{p}_{k,g,t} - \text{stopgrad}(\hat{p}_{k,g,t})}{P_{\text{scale}}} \|_2^2$, where $P_{\text{scale}} = \max_{k,g,t} p_{k,g,t}$, where stopgrad means skip the gradient in the back-propagation training. The purpose of stopping the gradient is to encourage the neural prediction to approach the feasible repaired solution while avoiding unnecessary differentiation through the piecewise-differentiable projection and bisection operations. This projection-alignment loss treats the repaired dispatch as a feasible target and reduces the discrepancy between the raw prediction $\widetilde{p}_{g,t}$ and the projected solution $\widehat{p}_{g,t}$, thereby decreasing the magnitude of the required repair.

**Bound-feasibility Loss.** As discussed in Section 3.2, the bound-feasibility loss $\mathcal{L}_{\text{bounds}}$ penalizes cases in which the demand lies outside the dynamically reachable interval and propagates corrective gradients to earlier dispatch decisions. We further introduce a pre-repair demand-balance loss $\mathcal{L}_{\text{bal}} = \frac{1}{NT} \sum_{k=1}^{N} \sum_{t=1}^{T} \| \frac{D_{k,t} - \sum_g \widetilde{p}_{k,g,t}}{D_{\text{scale}}} \|_2^2$, $D_{\text{scale}} = \max(D_{k,t}, \varepsilon)$, which encourages the neural prediction itself to satisfy the demand-balance constraint before projection. The overall training objective is

$$\mathcal{L} = \lambda_{\text{cost}} \mathcal{L}_{\text{cost}} + \lambda_{\text{proj}} \mathcal{L}_{\text{proj}} + \lambda_{\text{bal}} \mathcal{L}_{\text{bal}} + \lambda_{\text{bounds}} \mathcal{L}_{\text{bounds}}, \tag{6}$$

where the coefficients $\lambda_{\text{cost}}$, $\lambda_{\text{proj}}$, $\lambda_{\text{bal}}$, and $\lambda_{\text{bounds}}$ balance economic optimality and feasibility during training.

### 3.4 Complete Algorithm Procedure

We introduce the overall full framework in Algorithm 1 with the sub-algorithms inside the pipeline in the Appendix A for the interested readers.

**Algorithm 1** SMOP: Self-Supervised Multi-Period Optimization Proxy

**Input:** Dataset $\mathcal{D}^N = \{(\boldsymbol{u}^k, \boldsymbol{v}^k, \boldsymbol{w}^k, \boldsymbol{D}^k, \boldsymbol{R}^k)\}_{k=1}^N, P^{\min}, P^{\max}, RU, RD, SU, SD, \boldsymbol{p}_0, \boldsymbol{u}_0, \boldsymbol{c}$; network $f_\theta$; epochs $E$; batch size $B$.
$(\boldsymbol{L}^r, \boldsymbol{U}^r) \leftarrow$ BackwardForwardBoundPropagation($\mathcal{D}^N$) using Algorithm 2.
**for** $e = 1, \ldots, E$ **do**
  **for** each mini-batch $\mathcal{D}^B \subset \mathcal{D}^N$ **do**
    $(\widehat{\boldsymbol{p}}, \widetilde{\boldsymbol{p}}, \boldsymbol{s}^{\text{short}}, \boldsymbol{s}^{\text{spill}}) \leftarrow$ ForwardRepair($f_\theta, \boldsymbol{L}^r, \boldsymbol{U}^r, \mathcal{D}^B$).
    $\mathcal{L} \leftarrow$ Loss($\widehat{\boldsymbol{p}}, \widetilde{\boldsymbol{p}}, \boldsymbol{s}^{\text{short}}, \boldsymbol{s}^{\text{spill}}; \mathcal{D}^B$).
    Update $\theta$ using a gradient-based optimizer.
  **end for**
**end for**
**return** $\widehat{\boldsymbol{p}}$ for the target instances using the trained $f_\theta$ and ForwardRepair.

# 4 THEORETICAL ANALYSIS

The theoretical analysis establishes the effectiveness of the key components of the proposed SMOP framework. We first show that the forward–backward bound propagation and sequential forward recovery construct full horizon feasible intervals. We then derive the condition for the existence of a demand-balanced feasible dispatch within the propagated bounds and prove the uniqueness of the projection-based repair solution. Finally, we establish the feasibility of the proportional reserve allocation, which justifies computing reserve analytically after obtaining a feasible energy dispatch rather than learning $r_{g,t}$ jointly with $p_{g,t}$. The detailed proofs are presented in Appendix B.

**Proposition 1** (Full-Horizon Feasibility of Sequential Recovery)**.** *Let $[L^r_{g,t}, U^r_{g,t}]$ be the ramp-aware interval obtained by the sequential forward recovery in Section 3.2. Suppose $p_{g,0}$ is feasible and for every $t$ $L^r_{g,t} \le U^r_{g,t}, \sum_g L^r_{g,t} \le D_t \le \sum_g U^r_{g,t}$. If the projection repair in Section 3.2 returns $p_{g,t} \in [L^r_{g,t}, U^r_{g,t}], \sum_g p_{g,t} = D_t$, then the sequentially recovered dispatch is feasible over the full horizon, satisfying power balance, generation limits, and ramp-up and ramp-down constraints for all periods.*

The proof of Proposition 1 shows that, if sequential forward recovery initialized by backward–forward bound propagation produces nonempty intervals $[L^r_{g,t}, U^r_{g,t}]$, then any dispatch within these intervals satisfies the generator limits and ramping constraints. If $L^r_{g,t} \le U^r_{g,t}, \sum_g L^r_{g,t} \le D_t \le \sum_g U^r_{g,t}$ and the projection repair further finds $\hat{p}_{g,t} \in [L^r_{g,t}, U^r_{g,t}]$ with $\sum_g \hat{p}_{g,t} = D_t$, full-horizon feasibility is guaranteed. When $D_t \notin [\sum_g L^r_{g,t}, \sum_g U^r_{g,t}]$, Proposition 4 provides a corrective learning strategy by penalizing shortage and over-generation in the loss function.

**Proposition 2** (Feasibility of Proportional Reserve Allocation)**.** *As the effective reserve-capable upper bound is $r_{g,t} \le P^{\max}_g u_{g,t} - p_{g,t}$, suppose that, at each period $t$, $\sum_g p_{g,t} = D_t, 0 \le p_{g,t} \le P^{\max}_g u_{g,t}$, and the given commitment is feasible, satisfying the condition that $D_t + R_t \le \sum_g (P^{\max}_g u_{g,t})$. Then the proportional headroom allocation by $r_{g,t} = \frac{P^{\max}_g u_{g,t} - p_{g,t}}{\sum_j (P^{\max}_g u_{g,t} - p_{j,t})} R_t$ ensures the reserve feasibility, including $r_{g,t} \ge 0, p_{g,t} + r_{g,t} \le P^{\max}_g u_{g,t}$ and $\sum_g r_{g,t} = R_t$.*

This proposition shows that reserve does not need to be learned or repaired iteratively. Once the energy dispatch is feasible, reserve feasibility follows analytically from the remaining headroom, provided the commitment itself has sufficient reserve capacity.

**Proposition 3** (Analytic Solution of Feasible Projection)**.** *Let the feasible region be $\mathcal{F}_t = \{p_{g,t} : \sum_{g=1}^G p_{g,t} = D_t, L_{g,t} \le p_{g,t} \le U_{g,t}\}$, define the feasible projection is $\widehat{p}_{g,t} = \arg\min_{p_{g,t} \in \mathcal{F}_t} \frac{1}{2} \sum_{g=1}^G (p_{g,t} - \widetilde{p}_{g,t})^2$, then the analytic solution is $\widehat{p}_{g,t} = \mathrm{clip}(\widetilde{p}_{g,t} - \lambda_t, L_{g,t}, U_{g,t}) = \min(U_{g,t}, \max(L_{g,t}, p_{g,t}))$, where $\lambda_t$ satisfies $\sum_g \mathrm{clip}\,(\widetilde{p}_{g,t} - \lambda_t, L_{g,t}, U_{g,t}) = D_t$. Meanwhile, this is the unique Euclidean projection of $\widetilde{p}_t$ onto $\mathcal{F}_t$.*

The proposition derives an analytic solution for the projection-based repair layer, avoiding the need to solve an expensive optimization problem.

**Proposition 4** (Corrective Gradient of the Bound Feasibility Loss)**.** *Given the period-$t$ reachable bounds depend on the previous dispatch as $L^r_{g,t} = L^r_{g,t}(p_{t-1}), U^r_{g,t} = U^r_{g,t}(p_{t-1})$, and the bound-feasibility loss $\mathcal{L}_{bounds} = \max_t \ell_t = \ell^*$, and $\ell^* = c_t([D_t - \sum_g U^r_{g,t}]_+ + [\sum_g L^r_{g,t} - D_t]_+), c_t = \frac{1}{\max(D_t, \varepsilon)} > 0$. At any differentiable point, if $D_t > \sum_g U^r_{g,t}$ and the active upper bound is ramp-dependent, $U^r_{g,t} = p_{g,t-1} + \Phi_{g,t}$, then $\frac{\partial \ell^*}{\partial p_{g,t-1}} = -c_t < 0$. Hence, the negative-gradient direction increases $p_{g,t-1}$, enlarges the reachable upper bound, and locally reduces the shortage.*

*Similarly, if $\sum_g L^r_{g,t} > D_t$ and the active lower bound is ramp-dependent, then $\frac{\partial \ell^*}{\partial p_{g,t-1}} = c_t > 0$, so the negative-gradient direction decreases $p_{g,t-1}$, lowers the reachable lower bound, and locally reduces the spill. Therefore, the gradient in the back-propagation provides a corrective learning signal that moves the reachable interval toward containing $D_t$. Note if a bound is determined by a fixed generator limit, its derivative with respect to $p_{t-1}$ is zero.*

**Computational complexity** The backward-forward interval propagation and analytical reserve recovery require $\mathcal{O}(TG)$ operations. At each period, the exact bounded balance projection reduces to a one-dimensional root-finding problem, requiring $\mathcal{O}(G \log(1/\epsilon_{\mathrm{proj}}))$ operations using bisection and $\mathcal{O}(TG \log(1/\epsilon_{\mathrm{proj}}))$ over the full horizon, where $\epsilon_{\mathrm{proj}}$ is a small projection tolerance. Consequently, the proposed feasibility-preserving operations scale nearly linearly with both the number of resources and the scheduling horizon and require no large-scale Karush–Kuhn–Tucker (KKT) solve, which norminally cost $\mathcal{O}((TG)^3)$.

## 5 Related Work

**Differentiable Optimization and Repair.** Differentiable optimization layers such as OptNet and CVXPYLayers are useful to build up repair layers for feasibility (Amos & Kolter, 2017; Agrawal et al., 2019). Our approach differs fundamentally in how feasibility and differentiation are handled. Differentiable optimization typically embeds the constrained optimization problem as a whole into the learning architecture and differentiates through its optimality conditions, requiring the simultaneous treatment of all coupled constraints and the solution of potentially large KKT systems during the forward and backward passes. On the contrary, we exploit the specific resource–temporal structure of multi-period dispatch to decompose feasibility according to the interactions among constraints.

**Self-supervised Proxy.** End-to-End Learning and Repair (E2ELR) proxy is one of the typical self-supervised proxy for economic dispatch problems (Chen et al., 2023). E2ELR addresses *single-period* economic dispatch and restores power balance through a closed-form repair layer that proportionally moves generator outputs toward their *static* lower or upper limits (Chen et al., 2023). Primal-Dual Learning similarly targets *single-period* optimization, combining proportional repair with jointly trained primal and dual networks in an augmented-Lagrangian-inspired framework without labels (Park & Van Hentenryck, 2023). In contrast, our setting is multi-period: the feasible interval $[L^r_{g,t}, U^r_{g,t}]$ depends on both past dispatches and future requirements and must be tightened to preserve intertemporal ramping feasibility over the full horizon. Moreover, rather than using a predetermined proportional repair rule, SMOP learns the allocation fraction within each feasible interval, allowing the network to distribute available flexibility across generators according to the operating-cost objective while maintaining feasibility.

## 6 Numerical Analysis

We validate SMOP on public benchmarks against four neural-network baselines using feasibility, optimality-gap, training-time, and inference-time metrics. Additional studies on reserve allocation, numerical precision, component configurations, and projection iterations are provided in Appendix D. All experiments are conducted on a MacBook Pro with an Apple M4 chip and 16,GB of memory.

**Datasets and Neural Networks** The dataset is generated from three public unit-commitment benchmarks `or-lib/G_0_1_w` with $G = 10, 50, 100$ respectively. The benchmarks contain the parameters required by the mixed-integer programming (MIP) formulation (Morales-España et al., 2013), while the reference solutions used for evaluation are obtained by solving each instance with the HiGHS MIP solver (Huangfu & Hall, 2018). The systems contain $G$ generating units over a $T = 24$-hour horizon. We obtain $N = 1720$ feasible samples for each system, of which $90\%$ are used for training and the remainder for testing. We additionally generate $N$ feasible samples with nonzero reserve requirements to validate the proportional reserve-allocation method in Appendix D.1. SMOP achieves zero maximum constraint violation, a mean absolute optimality gap of $0.40\% \pm 0.003\%$, and an inference time of $0.83 \pm 0.02$ ms per sample, while the post-training proportional allocation satisfies the corresponding generator capacity and ramping constraints. Each sample contains a time-varying load profile. Specifically, the ratio between two consecutive demands, $\gamma_t = D_{t+1}/D_t$, is randomly generated from a Gaussian distribution chosen to approximate realistic utility load variations (Xavier et al., 2021). The resulting demands range from $568.11$ MW to $1142.2$ MW, with a standard deviation of $104.8$ MW. These variations capture changes in daily load shape and peak demand, making the scheduling problem challenging. Only instances solved to an optimality gap below $10^{-4}$ are retained, with feasible commitment schedules satisfying $D_t + R_t \leq \sum_g P^{\max}_g u_{g,t}, \forall t$.

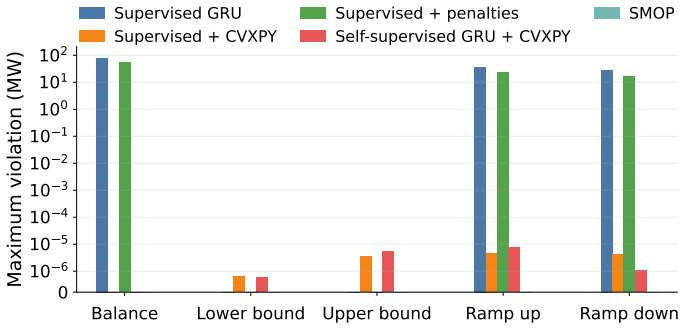


Figure 2: Comparison of maximum violation of constraints

Our bidirectional GRU processes a $(13G+2)$-dimension feature vector at each time period using one recurrent layers, each with hidden size $H = 32$. The concatenated forward and backward hidden states therefore have dimension $2H = 64$. The prediction head consists of two fully connected layers with GELU (Gaussian Error Linear Units) activations (Hendrycks & Gimpel, 2016): the first maps the representation from $2H$ to $H$, and the second maps it from $H$ to $G$, producing one output score for each generator. GELU is adopted to provide a smooth nonlinear mapping for gradient-based training. We adapt the AdamW (Loshchilov & Hutter, 2017) optimizer to train neural network with

batch size $B = 32$ and learning rate being $10^{-3}$ and different random seed. The weights for the loss function are $\lambda_{\text{cost}} = 1.0, \lambda_{\text{proj}} = 1.0, \lambda_{\text{bal}} = 0.1, \lambda_{\text{bounds}} = 0.5$.

**Evaluation Metrics** We evaluate the methods from three aspects: constraint feasibility, solution quality, and computational efficiency. Table 3 in the Appendix C summarizes the main metrics. Maximum violations are used as the primary feasibility measures, as they indicate whether all predicted solutions satisfy the corresponding constraints. Solution quality is evaluated by the relative optimality gap. Computational efficiency is measured by training and inference time.

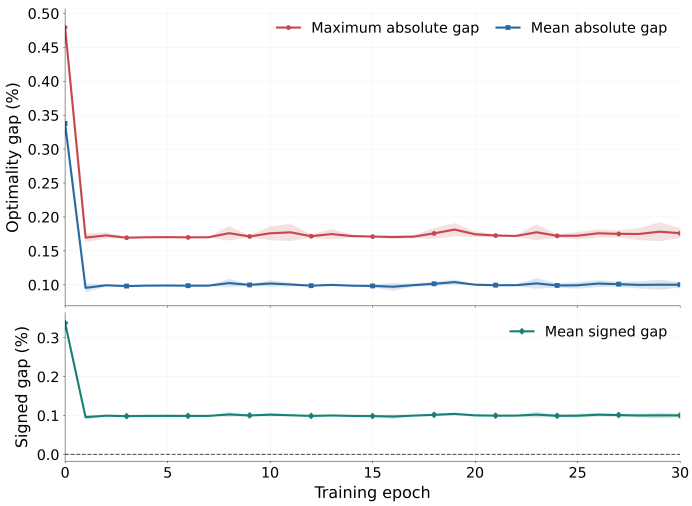


Figure 3: Optimality Gap Learning Curves

**Baselines** We compare SMOP with four baselines. The first is a purely supervised method (`Supervised GRU`) using the same GRU architecture but without any projection or repair layer. Its dispatch prediction is given by $p_{g,t} = P^{\min}_{g,t} u_{g,t} + \eta_{g,t} \left( P^{\max}_{g,t} u_{g,t} - P^{\min}_{g,t} u_{g,t} \right)$, where $\eta_{g,t}$ is trained solely using labeled optimal solutions. The second baseline (`Supervised + penalties`) also adopts the same GRU backbone but augments the prediction with penalties designed to reduce violations of the operational constraints during training. We further construct two baselines based on differentiable programming (DP), implemented using CVXPYLayers (Agrawal et al., 2019). Both employ the same GRU model, while the predicted dispatch is projected onto the feasible region by solving $\hat{p} = \arg\min_{p \in \mathcal{C}} \sum_{g,t} \left( p_{g,t} - \tilde{p}_{g,t} \right)^2$, where $\mathcal{C}$ denotes the feasible set defined by constraints equation 2–equation 5. For the supervised DP baseline (`Supervised + CVXPY`), the GRU is trained with labeled solutions in the same manner as the supervised model, while the optimization projection is applied only at inference. For the self-supervised DP baseline (`Self-supervised GRU + CVXPY`), the projection layer is included during training and no labels are required; the loss is composed of $\mathcal{L} = \mathcal{L}_{\text{cost}} + \mathcal{L}_{\text{proj}}$, where $\mathcal{L}_{\text{cost}}$ promotes economic optimality and $\mathcal{L}_{\text{proj}}$ penalizes the correction between the raw neural prediction and its feasible projection.

### 6.1 Performance Comparison on Feasibility

Figure 2 compares the maximum violations of the power-balance, generation lower- and upper-bound, ramp-up, and ramp-down constraints, where missing bars indicate zero violation. The "Supervised GRU" and "Supervised + penalty" baselines exhibit relatively large violations in power balance and ramping constraints. Their zero violations of the generation bounds mainly result from parameterizing the neural-network output as a proportion within the admissible generation range. In contrast, SMOP achieves zero power-balance violation, generation limits and ramping constraints, demonstrating substantially stronger feasibility than the competing baselines.

### 6.2 Performance on Optimality

SMOP reduces the optimality gap by directly minimizing generation cost during training. Figure 3 reports the mean optimality gap ( $\frac{1}{N} \sum_b |\delta_b|$ where $\delta_b$ is the relative optimality gap ) of training the ORLIB-10 unit datasets with different random seeds, achieves a 70.36% reduction, and maximum absolute optimality gaps ($\max_b |\delta_b|$), a 63.27% reduction over training epochs. Both metrics exhibit an overall downward trend and eventually converge, with the mean absolute optimality gap falling below 0.1%.

### 6.3 Evaluation on Scalability with Overall Capability

[2]"Below Tolerance" indicates that the measured value falls below the numerical tolerance and is treated as zero; such values may arise from floating-point round-off.

Table 1: Comparison of dispatch-learning methods (mean ± standard deviation).

| Method | Maximum violation (MW) | Absolute gap (%) | Inference time (ms/sample) | Training time (s) |
|---|---|---|---|---|
| Supervised | $57.76 \pm 8.50$ | $0.76 \pm 0.27$ | $\mathbf{0.06 \pm 0.002}$ | $36.59 \pm 0.23$ |
| Supervised + penalties | $62.24 \pm 20.27$ | $0.85 \pm 0.22$ | $0.07 \pm 0.01$ | $\mathbf{36.13 \pm 0.80}$ |
| Supervised + CVXPY | $\mathbf{(4.36 \pm 2.14) \times 10^{-6}}$ | $\mathbf{0.03 \pm 0.01}$ | $2.90 \pm 0.15$ | $36.59 \pm 0.23$ |
| Self-supervised + CVXPY | $(3.54 \pm 1.57) \times 10^{-6}$ | $0.62 \pm 0.00$ | $3.59 \pm 0.06$ | $174.4 \pm 0.73$ |
| SMOP | Below Tolerance[2] | $\mathbf{0.098 \pm 0.0008}$ | $\mathbf{0.80 \pm 0.004}$ | $\mathbf{78.71 \pm 0.98}$ |

Table 2: SMOP scaling over three seeds (mean ± standard deviation).

| Dataset | Max. viol. (MW) | Abs. gap (%) | Train (s) | Infer. (ms/sample) | Solver. (ms/sample) | Speedup times |
|---|---|---|---|---|---|---|
| OR-LIB 10 | 0.0 | $0.098 \pm 0.0008$ | $86.1 \pm 1.0$ | $0.80 \pm 0.004$ | $7.78 \pm 0.22$ | **9.68**× |
| OR-LIB 50 | 0.0 | $0.45 \pm 0.01$ | $209.7 \pm 1.6$ | $1.39 \pm 0.04$ | $38.9 \pm 3.67$ | **27.81**× |
| OR-LIB 100 | 0.0 | $0.38 \pm 0.03$ | $399.6 \pm 17.0$ | $1.90 \pm 0.12$ | $81.08 \pm 7.85$ | **42.58**× |

We compare the five methods in two groups in terms of maximum constraint violation, mean absolute optimality gap, inference time, and training time, as summarized in Table 1. The first three methods supervised approaches trained with labeled solutions. Among them, "Supervised + CVXPY" achieves relatively small constraint violations, but at the cost of substantially higher inference and training times. The last two methods are trained in a self-supervised manner. SMOP achieves a smaller mean absolute optimality gap together with faster inference and training, while maintaining zero constraint violations MW or smaller are numerical round-off errors and are treated as zero.). Overall, SMOP provides the most favorable trade-off, with an inference time $0.88$ ms ($9.68\times$ speedup than HiGHS solver), $2.27 \times 10^{-13}$ MW maximum constraint violation, and the smallest optimality gap among the compared methods. In addition, we evaluate SMOP under seven configurations in Appendix D.3, showing that backward–forward bound propagation and sequential forward recovery are the most critical components for preserving feasibility and optimality. Also, we study the sensitivity to numerical precision and the number of bisection iterations. Results in Appendix D.2 show that `float64` with at least 20 iterations effectively eliminates numerical residuals while maintaining low computational cost.

To evaluate scalability, we test SMOP on three OR-LIB datasets with $G = 10, 50$, and $100$ generators, as reported in Table 2. SMOP maintains essentially zero constraint violations and small optimality gaps as the number of generators increases by an order of magnitude. Although training time grows with problem size, inference remains at the millisecond level. We compare the per-instance solution time of SMOP and the HiGHS linear-programming solver, showing speedups ranging from $9.68\times$ to $42.58\times$. A key source of this acceleration is the batch-processing capability of neural networks, which allows many instances to be evaluated simultaneously and substantially reduces the amortized inference time per instance.

## 7 Conclusions and Future Work

**Contributions.** We propose SMOP, a self-supervised multi-period optimization proxy for resource allocation with temporal constraints. SMOP integrates backward–forward interval propagation, projection-based balance repair, and sequential forward bound recovery to efficiently preserve full-horizon feasibility. We establish theoretical feasibility guarantees and the associated conditions, while experiments demonstrate zero constraint violations, less than 0.5% optimality gaps, millisecond-level inference, and up to $42.58\times$ speedup over the conventional solver.

**Limitations.** SMOP targets multi-period resource-allocation problems with period-wise balance constraints and resource-wise temporal coupling. Fixed-commitment economic dispatch serves as a representative application combining system-wide balance with unit-level temporal constraints. While the theoretical results apply to the formulated problem class, the current experiments are limited to economic-dispatch benchmarks; evaluation on production planning and other temporally coupled resource-allocation problems is left for future work. In addition, SMOP currently considers linear models with fixed commitment and basic reserve constraints; extensions to nonlinear models, endogenous commitment decisions, and temporally coupled reserve constraints remain future directions.

## REFERENCES


Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Tiziano Bacci, Antonio Frangioni, and Claudio Gentile. Start-up/shut-down MINLP formulations for the unit commitment with ramp constraints. In *Graphs and Combinatorial Optimization: from Theory to Applications – CTW 2020 Proceedings*, volume 5 of *AIRO Springer Series*, pp. 335–347. Springer, Cham, 2021.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004. ISBN 9780521833783.

Stephen Boyd, Steven Diamond, Kwangmoo Koh, Peter Nystrup, Enzo Busseti, Ronald N Kahn, and Jan Speth. Multi-period trading via convex optimization. *Foundations and Trends in Optimization*, 3(1):1–76, 2017.

Wenbo Chen, Seonho Park, Mathieu Tanneau, and Pascal Van Hentenryck. Learning optimization proxies for large-scale security-constrained economic dispatch. *Electric Power Systems Research*, 213:108566, 2022. doi: 10.1016/j.epsr.2022.108566.

Wenbo Chen, Mathieu Tanneau, and Pascal Van Hentenryck. End-to-end feasible optimization proxies for large-scale economic dispatch. *IEEE Transactions on Power Systems*, 39(2):4723–4734, 2023.

Priya L Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. *International Conference on Learning Representations*, 2021.

Antonio Frangioni and Claudio Gentile. Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research*, 54(4):767–775, 2006.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

Krzysztof C. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112(2):473–491, 2008. doi: 10.1007/s10107-006-0050-z.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Germán Morales-España, Jesus M. Latorre, and Andres Ramos. Tight and compact milp formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, 28(4):4897–4908, 2013.

Hoang Nguyen and Priya Donti. Fsnet: Feasibility-seeking neural network for constrained optimization with guarantees. *Advances in neural information processing systems*, 38:39670–39708, 2025.

Seonho Park and Pascal Van Hentenryck. Self-supervised primal-dual learning for constrained optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4052–4060, 2023. doi: 10.1609/aaai.v37i4.25520.

Seonho Park and Pascal Van Hentenryck. Self-supervised learning for large-scale preventive security constrained dc optimal power flow. *IEEE Transactions on Power Systems*, 40(3):2205–2216, 2024.

Kevin Wu, Rabab Haider, and Pascal Van Hentenryck. Trust-region benders for scalable multiperiod expansion planning for transmission and storage. In *Proceedings of the PowerUp Conference*, volume 1, 2026.

Álinson S Xavier, Feng Qiu, and Shabbir Ahmed. Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756, 2021.

**Algorithm 2** Backward–Forward Bound Propagation

**Require:** Given $\{(\boldsymbol{u}^k, \boldsymbol{v}^k, \boldsymbol{w}^k, D^k, R^k)\}_{k=1}^N$, and ramp-limit matrices $\Phi = RU\boldsymbol{u} + SU\boldsymbol{v}, \Psi = RD\boldsymbol{u} + SD\boldsymbol{w} \in \mathbb{R}^{G\times T}$; demand $D$; generator limits $P^{\min}, P^{\max}$; initial dispatch $\boldsymbol{p}_0, \boldsymbol{u}_0$; maximum number of sweeps $N_{\mathrm{bf}}$; tolerance $\epsilon$.

$L^B_{:,0} \leftarrow \boldsymbol{p}_0$
$U^B_{:,0} \leftarrow \boldsymbol{p}_0$
$n \leftarrow 0; \quad \delta_{\max} \leftarrow +\infty$
**while** $n < N_{\mathrm{bf}}$ **and** $\delta_{\max} > \epsilon$ **do**
▷ Forward propagation
  $L^B_{:,1} \leftarrow \max\{L^B_{:,1}, L^B_{:,0} - \Psi_{:,1}\}$
  $U^B_{:,1} \leftarrow \min\{U^B_{:,1}, L^B_{:,0} + \Phi_{:,1}\}$
  **for** $t = 2$ to $T$ **do**
    $L^B_{:,t} \leftarrow \max\{L^B_{:,t}, L^B_{:,t-1} - \Psi_{:,t}\}$
    $U^B_{:,t} \leftarrow \min\{U^B_{:,t}, U^B_{:,t-1} + \Phi_{:,t}\}$
  **end for**
▷ Backward propagation
  **for** $t = T-1$ downto 1 **do**
    $L^B_{:,t} \leftarrow \max\{L^B_{:,t}, L^B_{:,t+1} - \Psi_{:,t}\}$
    $U^B_{:,t} \leftarrow \min\{U^B_{:,t}, U^B_{:,t+1} + \Phi_{:,t}\}$
  **end for**
▷ Load-balance tightening: Jacobi update
  $\boldsymbol{s}^U \leftarrow \mathbf{1}_G^\top U^B$
  $\boldsymbol{s}^L \leftarrow \mathbf{1}_G^\top L^B$
  $L^+ \leftarrow \max\{L^B, \mathbf{1}_G D^\top - \mathbf{1}_G \boldsymbol{s}^U + U^B\}$
  $U^+ \leftarrow \min\{U^B, \mathbf{1}_G D^\top - \mathbf{1}_G \boldsymbol{s}^L + L^B\}$
  $(L^B, U^B) \leftarrow (L^+, U^+)$
  **if** $\mathrm{any}(L^B > U^B)$ **then**
    **return** INFEASIBLE
  **end if**
  $\delta_{\max} \leftarrow \max\{\|L^B - L^{\mathrm{old}}\|_\infty, \|U^B - U^{\mathrm{old}}\|_\infty\}$
  $n \leftarrow n + 1$
**end while**
**return** $(L^B, U^B)$

# A Detailed Algorithms

# B Theoretical Proofs

**Proposition 1** (Full-Horizon Feasibility of Sequential Recovery) Let $[L^r_{g,t}, U^r_{g,t}]$ be the ramp-aware interval obtained by the sequential forward recovery in Section 3.2. Suppose $p_{g,0}$ is feasible and, for every $t$, $L^r_{g,t} \le U^r_{g,t}, \sum_g L^r_{g,t} \le D_t \le \sum_g U^r_{g,t}$. If the projection repair in Section 3.2 returns $p_{g,t} \in [L^r_{g,t}, U^r_{g,t}], \sum_g p_{g,t} = D_t$, then the sequentially recovered dispatch is feasible over the full horizon, satisfying power balance, generation limits, and ramp-up and ramp-down constraints for all periods.

*Proof.* Given the conditions $[L^r_{g,t}, U^r_{g,t}]$ and $L^r_{g,t} \le U^r_{g,t}, \sum_g L^r_{g,t} \le D_t \le \sum_g U^r_{g,t}$, the intersection $\{\boldsymbol{p}_t : L^r_{g,t} \le p_{g,t} \le U^r_{g,t},\ \forall g\} \cap \{\boldsymbol{p}_t : \sum_g p_{g,t} = D_t\}$ is nonempty. Therefore, the projection repair can produce a balanced dispatch satisfying $p_{g,t} \in [L^r_{g,t}, U^r_{g,t}], \sum_g p_{g,t} = D_t$. By construction of the recovered bounds, $L^r_{g,t} \ge \max\{P^{\min}_g u_{g,t}, p_{g,t-1} - \Psi_{g,t}\}$, and $U^r_{g,t} \le \min\{P^{\max}_g u_{g,t}, p_{g,t-1} + \Phi_{g,t}\}$. Thus, any $p_{g,t} \in [L^r_{g,t}, U^r_{g,t}]$ satisfies $P^{\min}_g u_{g,t} \le p_{g,t} \le P^{\max}_g u_{g,t}$, as well as $p_{g,t} - p_{g,t-1} \le \Phi_{g,t}, p_{g,t-1} - p_{g,t} \le \Psi_{g,t}$. Hence, generation limits and ramp-up/down constraints are satisfied at period $t$, while the projection enforces power balance. Starting from the feasible initial dispatch $p_{g,0}$, the sequential recovery applies the same construction at each subsequent period using the realized dispatch $p_{g,t-1}$. Therefore, by induction over $t = 1, \ldots, T$, the recovered trajectory satisfies power balance, generation limits, and ramping constraints over the entire horizon. □

**Algorithm 3** Box–Balance Projection Repair

**Require:** Predicted dispatch $\boldsymbol{p}$, lower and upper bounds $L, U$, demand $D$, maximum bisection iterations $N_b$, and tolerance $\epsilon_{\text{proj}}$
**Ensure:** Repaired dispatch $\boldsymbol{p}$ and multiplier $\lambda$

$\lambda^{\text{lo}} \leftarrow \min_{g\in\mathcal{G}}(p_g - U_g)$
$\lambda^{\text{hi}} \leftarrow \max_{g\in\mathcal{G}}(p_g - L_g)$
**for** $k = 1$ to $N_b$ **do**
  $\lambda \leftarrow \frac{1}{2}\left(\lambda^{\text{lo}} + \lambda^{\text{hi}}\right)$
  $p_g \leftarrow \text{clip}(x_g - \lambda, L_g, U_g)$  for all $g \in \mathcal{G}$
  **if** $\left|\sum_g p_g - \widehat{D}\right| \le \epsilon_{\text{proj}}$ **then**
    **return** $(p, \lambda)$
  **else if** $\sum_g p_g > \widehat{D}$ **then**
    $\lambda^{\text{lo}} \leftarrow \lambda$
  **else**
    $\lambda^{\text{hi}} \leftarrow \lambda$
  **end if**
**end for**
$\lambda \leftarrow \frac{1}{2}\left(\lambda^{\text{lo}} + \lambda^{\text{hi}}\right)$
$p_g \leftarrow \text{clip}(x_g - \lambda, L_g, U_g)$  for all $g \in \mathcal{G}$
**return** $(\boldsymbol{p}, \lambda)$

**Algorithm 4** Sequential Forward Recovery

**Require:** $\mathcal{D}^B = \{(\boldsymbol{u}^k, \boldsymbol{v}^k, \boldsymbol{w}^k, \boldsymbol{D}^k, \boldsymbol{R}^k)\}_{k=1}^B, P^{\min}, P^{\max}, RU, RD, SU, SD, \boldsymbol{p}_0, \boldsymbol{u}_0, \boldsymbol{c}$, ramp-limit matrices $\Phi = RU\boldsymbol{u} + SU\boldsymbol{v}, \Psi = RD\boldsymbol{u} + SD\boldsymbol{w}$; global bounds $(L^B, U^B)$; neural networks $f_\theta$.
**Ensure:** Repaired dispatch $\hat{\boldsymbol{p}}$ and balance slacks

$\hat{p}_{g,0} \leftarrow p_{g,0}$ and $\hat{u}_{g,0} \leftarrow u_{g,0}$ for all $g \in \mathcal{G}$
**for** $t = 1$ to $T$ **do**  ▷ Construct forward-repair bounds
  $L^r_{:,t} \leftarrow \max\left\{L^B_{:,t},\ P^{\min}\boldsymbol{u}_t,\ \hat{\boldsymbol{p}}_{t-1} - \Psi_{:,t-1}\right\}$
  $U^r_{:,t} \leftarrow \min\left\{U^B_{:,t},\ P^{\max}\boldsymbol{u}_t,\ \hat{\boldsymbol{p}}_{t-1} + \Phi_{:,t-1}\right\}$
  **if** $L^r_{g,t} > U^r_{g,t}$ for any $g \in \mathcal{G}$ **then**
    **return** INFEASIBLE
  **end if**
  $\boldsymbol{\eta}_t \leftarrow f_\theta(\mathcal{D}^B)$  ▷ $0 \le \eta_{g,t} \le 1$
  $\tilde{\boldsymbol{p}}_t \leftarrow L^r_{:,t} + \boldsymbol{\eta}_t \odot \left(U^r_{:,t} - L^r_{:,t}\right)$  ▷ Repair demand using balance slack when necessary
  $s^{\text{short}}_t \leftarrow \left[D_t - \sum_g U^r_{g,t}\right]_+$
  $s^{\text{spill}}_t \leftarrow \left[\sum_g L^r_{g,t} - D_t\right]_+$
  $\widehat{D}_t = \min(\sum_g U^r_{g,t}, \max(D_t, \sum_g L^r_{g,t}))$
  ▷ Repair by the Algorithm 3
  Find $\lambda_t$ such that $\sum_g \text{clip}(\tilde{p}_t - \lambda_t, L^r_t, U^r_t) = \widehat{D}_t$ by Algorithm 3
  $\hat{p}_t \leftarrow \text{clip}(\tilde{p}_t - \lambda_t, L^r_t, U^r_t)$
**end for**
**return** $(\hat{\boldsymbol{p}}, \tilde{\boldsymbol{p}}, \boldsymbol{s}^{\text{short}}, \boldsymbol{s}^{\text{spill}})$

**Proposition 2** (Feasibility of Proportional Reserve Allocation) As the effective reserve-capable upper bound is $r_{g,t} \le P^{\max}_g u_{g,t} - p_{g,t}$, suppose that, at each period $t$, $\sum_g p_{g,t} = D_t, 0 \le p_{g,t} \le \bar{P}^{\max}_g u_{g,t}$, and the given commitment is feasible, satisfying the condition that $D_t + R_t \le \sum_g (P^{\max}_g u_{g,t})$. Then the proportional headroom allocation by $r_{g,t} = \frac{\bar{P}^{\max}_g u_{g,t} - p_{g,t}}{\sum_j (P^{\max}_g u_{g,t} - p_{j,t})} R_t$ ensures the reserve feasibility, including $r_{g,t} \ge 0, p_{g,t} + r_{g,t} \le \bar{P}^{\max}_g u_{g,t}$ and $\sum_g r_{g,t} = R_t$.

*Proof.* Under the basic reserve model, the reserve of generator $g$ at time $t$ is bounded by the remaining committed capacity, i.e., $0 \le r_{g,t} \le h_{g,t}$, where the headroom is $h_{g,t} = P_g^{\max}\bar{u}_{g,t} - \hat{p}_{g,t}$, the available headroom is nonnegative: $h_{g,t} \ge 0$. Moreover, using $\sum_g p_{g,t} = D_t$, the aggregate headroom is $H_t = \sum_g h_{g,t} = \sum_g \bar{P}_g^{\max} u_{g,t} - \sum_g p_{g,t} = \sum_g \bar{P}_g^{\max} u_{g,t} - D_t \ge R_t$. Because $h_{g,t} \ge 0$, we have $r_{g,t} \ge 0$. Furthermore, $R_t/H_t \le 1$, and hence $r_{g,t} = \frac{R_t}{H_t} h_{g,t} \le h_{g,t}$. Therefore, $p_{g,t} + r_{g,t} \le p_{g,t} + h_{g,t} = \bar{P}_g^{\max} u_{g,t}$. Finally, $\sum_g r_{g,t} = \frac{R_t}{H_t} \sum_g h_{g,t} = R_t$. If additional reserve constraints, such as reserve ramping, zonal requirements, or unit-specific reserve limits, are imposed, then energy dispatch and reserve may need to be recovered jointly.

□

**Proposition 3** (Analytic Solution of Feasible Projection) Let the feasible region be $\mathcal{F}_t = \left\{p_{g,t} : \sum_{g=1}^G p_{g,t} = D_t, L_{g,t} \le p_{g,t} \le U_{g,t}\right\}$, define the feasible projection is $\widehat{p}_{g,t} = \arg\min_{p_{g,t}\in\mathcal{F}_t} \frac{1}{2}\sum_{g=1}^G (p_{g,t} - \widetilde{p}_{g,t})^2$, then the analytic solution is $\widehat{p}_{g,t} = \mathrm{clip}(\widetilde{p}_{g,t} - \lambda_t, L_{g,t}, U_{g,t}) = \min(U_{g,t}, \max(L_{g,t}, p_{g,t}))$, where $\lambda_t$ satisfies $\sum_g \mathrm{clip}\left(\widetilde{p}_{g,t} - \lambda_t, L_{g,t}, U_{g,t}\right) = D_t$. Meanwhile, this is the unique Euclidean projection of $\widetilde{p}_t$ onto $\mathcal{F}_t$.

*Proof.* Because the objective is strictly convex and $\mathcal{F}_t$ is convex, if $\mathcal{F}_t \neq \emptyset$, the optimal dispatch $\widehat{p}_t$ is unique.

The feasible set is nonempty exactly when $\sum_g L_{g,t} \le D_t \le \sum_g U_{g,t}$. Introduce $\lambda_t$ for $\sum_g p_{g,t} = D_t, \alpha_{g,t} \ge 0$ for $L_{g,t} - p_{g,t} \le 0$, and $\beta_{g,t} \ge 0$ for $p_{g,t} - U_{g,t} \le 0$. The Lagrangian is $\mathcal{L} = \frac{1}{2}\sum_g (p_{g,t} - \widetilde{p}_{g,t})^2 + \lambda_t \left(\sum_g p_{g,t} - D_t\right) + \sum_g \alpha_{g,t}(L_{g,t} - p_{g,t}) + \sum_g \beta_{g,t}(p_{g,t} - U_{g,t})$. The stationary condition is $p_{g,t} - \widetilde{p}_{g,t} + \lambda_t - \alpha_{g,t} + \beta_{g,t} = 0$. Complementary slackness gives $\alpha_{g,t}(L_{g,t} - p_{g,t}) = 0, \beta_{g,t}(p_{g,t} - U_{g,t}) = 0$.

Consider the three possible cases: If $L_{g,t} < p_{g,t} < U_{g,t}$, then neither bound is active, so $\alpha_{g,t} = \beta_{g,t} = 0$. Stationarity gives $p_{g,t} = \widetilde{p}_{g,t} - \lambda_t$. If the lower bound is active, $p_{g,t} = L_{g,t}$. Then $\beta_{g,t} = 0$, and stationarity gives $\alpha_{g,t} = L_{g,t} - \widetilde{p}_{g,t} + \lambda_t \ge 0$. Hence $\widetilde{p}_{g,t} - \lambda_t \le L_{g,t}$. Therefore, $p_{g,t} = L_{g,t}$. Similarly, if the upper bound is active, $p_{g,t} = U_{g,t}$, and stationarity gives $\beta_{g,t} = \widetilde{p}_{g,t} - \lambda_t - U_{g,t} \ge 0$. Thus, $\widetilde{p}_{g,t} - \lambda_t \ge U_{g,t}$, and therefore $p_{g,t} = U_{g,t}$. Combining the three cases, $\widehat{p}_{g,t} = \mathrm{clip}\left(\widetilde{p}_{g,t} - \lambda_t, L_{g,t}, U_{g,t}\right)$, where $\mathrm{clip}(x, L, U) = \min\{U, \max\{L, x\}\}$.The $\lambda_t$ can be solved very efficiently by bisection, see details in Algorithm 3. □

**Proposition 4** (Corrective Gradient of the Bound Feasibility Loss) Given the period-$t$ reachable bounds depend on the previous dispatch as $L_{g,t}^r = L_{g,t}^r(p_{t-1}), U_{g,t}^r = U_{g,t}^r(p_{t-1})$, and the bound-feasibility loss $\mathcal{L}_{\text{bounds}} = \max_t \ell_t = \ell^*$, and $\ell^* = c_t([D_t - \sum_g U_{g,t}^r]_+ + [\sum_g L_{g,t}^r - D_t]_+), c_t = \frac{1}{\max(D_t, \varepsilon)} > 0$. At any differentiable point, if $D_t > \sum_g U_{g,t}^r$ and the active upper bound is ramp-dependent, $U_{g,t}^r = p_{g,t-1} + \Phi_{g,t}$, then $\frac{\partial \ell^*}{\partial p_{g,t-1}} = -c_t < 0$. Hence, the negative-gradient direction increases $p_{g,t-1}$, enlarges the reachable upper bound, and locally reduces the shortage.

Similarly, if $\sum_g L_{g,t}^r > D_t$ and the active lower bound is ramp-dependent, then $\frac{\partial \ell^*}{\partial p_{g,t-1}} = c_t > 0$, so the negative-gradient direction decreases $p_{g,t-1}$, lowers the reachable lower bound, and locally reduces the spill. Therefore, the gradient path $\theta \to p_{t-1} \to (L_t^r, U_t^r) \to \ell^*$ provides a corrective learning signal that moves the reachable interval toward containing $D_t$. If a bound is determined by a fixed generator limit, its derivative with respect to $p_{t-1}$ is zero.

*Proof.* Consider first the shortage case, $D_t > \sum_g U_{g,t}^r$. Then $\ell^* = c_t \|D_t - \sum_g U_{g,t}^r\|_\infty$, and therefore $\frac{\partial \ell^*}{\partial U_{g,t}^r} = -c_t$. If the active upper bound is ramp-dependent, $U_{g,t}^r = p_{g,t-1} + a_{g,t}$, then $\frac{\partial U_{g,t}^r}{\partial p_{g,t-1}} = 1$. By the chain rule, $\frac{\partial \ell^*}{\partial p_{g,t-1}} = \frac{\partial \ell^*}{\partial U_{g,t}^r}\frac{\partial U_{g,t}^r}{\partial p_{g,t-1}} = -c_t$. Hence the negative-gradient direction satisfies $-\frac{\partial \ell^*}{\partial p_{g,t-1}} = c_t > 0$, which tends to increase $p_{g,t-1}$. As long as the same branch remains active, $U_{g,t}^r$ increases by the same first-order amount, and thus $D_t - \sum_g U_{g,t}^r$ decreases.

For the spill case, $\sum_g L_{g,t}^r > D_t$, we have $\ell^* = c_t \left(\sum_g L_{g,t}^r - D_t\right)$, and therefore $\frac{\partial \ell^*}{\partial L_{g,t}^r} = c_t$. If $L_{g,t}^r = p_{g,t-1} - b_{g,t}$, then $\frac{\partial L_{g,t}^r}{\partial p_{g,t-1}} = 1$, which gives $\frac{\partial \ell^*}{\partial p_{g,t-1}} = c_t > 0$. The negative-gradient direction therefore decreases $p_{g,t-1}$, lowering the active reachable lower bound and reducing $\sum_g L_{g,t}^r - D_t$. Finally, since $p_{t-1} = p_{t-1}(\theta)$, the gradient propagates to the neural-network parameters according to $\nabla_\theta \ell^* = \left(\frac{\partial p_{t-1}}{\partial \theta}\right)^\top \nabla_{p_{t-1}} \ell^*$. At a locally differentiable

Table 3: Evaluation metrics. Here, $[x]_+ = \max(x, 0)$.

| Metric | Definition |
|---|---|
| Balance | $e^{\text{bal}}_{k,t} = \left\lvert \sum_g p_{k,g,t} - D_{k,t} \right\rvert, \quad e^{\text{bal}}_{\max} = \max_{k,t} e^{\text{bal}}_{k,t}, \quad e^{\text{bal}}_{\text{mean}} = \frac{1}{KT} \sum_{k,t} e^{\text{bal}}_{k,t}$ |
| Lower bounds | $e^{\text{lower}}_{\max} = \max_{k,g,t} \left[ P^{\min}_{g,t} u_{k,g,t} - p_{k,g,t} \right]_+,$ |
| Upper bounds | $e^{\text{upper}}_{\max} = \max_{k,g,t} \left[ p_{k,g,t} - P^{\max}_{g,t} u_{k,g,t} \right]_+$ |
| Ramp Up | $e^{\text{RU}}_{\max} = \max_{k,g,t} [p_{k,g,t} - p_{k,g,t-1} - \Phi_{k,g,t}]_+,$ |
| Ramp Down | $e^{\text{RD}}_{\max} = \max_{k,g,t} [p_{k,g,t-1} - p_{k,g,t} - \Psi_{k,g,t}]_+$ |
| Reserve shortage | $e^{\text{reserve}}_{\max} = \max_{k,t} [R_{k,t} - \sum_g H_{k,g,t}]_+, \quad H_{k,g,t} = P^{\max}_{g,t} u_{k,g,t} - p_{k,g,t}$ |
| Maximum Violation | $\max[e^{\text{bal}}_{\max}, e^{\text{lower}}_{\max}, e^{\text{upper}}_{\max}, e^{\text{RU}}_{\max}, e^{\text{RD}}_{\max}, e^{\text{reserve}}_{\max}]$ |
| Optimality gap | $\delta_k = 100 \frac{C_k(\hat{p}_{k,g,t}) - C^\star_k}{S_{\text{scale}}}, C_k(\hat{p}_{k,g,t})) = \sum_g \sum_t c_{g,t} \hat{p}_{k,g,t}$ |
| | $\text{MeanSignedGap} = \frac{1}{N} \sum_k \delta_k,$ |
| | $\text{MeanAbsGap} = \frac{1}{N} \sum_k \lvert\delta_k\rvert,$ |
| | $\text{MaxAbsGap} = \max_k \lvert\delta_k\rvert.$ |

Table 4: SMOP optimality and reserve feasibility over random seeds (mean $\pm$ standard deviation).

| Dataset | Absolute Gap (%) | $e^{\text{bal}}_{\max}$ (MW) | $e^{\text{reserve}}_{\max}$ (MW) |
|---|---|---|---|
| ORLIB 10 $R_t > 0$ | $0.42 \pm 3.61 \times 10^{-3}$ | $4.55 \times 10^{-13} \pm 0$ | $4.26 \times 10^{-14} \pm 0$ |

point, a sufficiently small gradient-descent step $\theta^+ = \theta - \eta \nabla_\theta \ell^*$ satisfies $\ell^*(\theta^+) = \ell^*(\theta) - \eta \|\nabla_\theta \ell^*\|_2^2 + o(\eta)$. Thus, whenever $\nabla_\theta \ell^* \neq 0$, there exists a sufficiently small $\eta > 0$ such that $\ell^*(\theta^+) < \ell^*(\theta)$. Therefore, minimizing the reachable-bound loss locally reduces the shortage or spill caused by differentiable ramp-dependent bounds. □

# C Detailed Metric

The definitions of metrics are presented in Table 3.

# D Details of Experimental Setup

## D.1 Performance of proportional reserve allocation method

We use 1,720 samples from the OR-LIB 10-unit benchmark, with 90% for training and 10% for validation, and set the reserve requirement to $R_t = 0.10 D_t$. SMOP is trained for 30 epochs using multiple random seeds. It achieves zero maximum constraint violation, a mean absolute optimality gap of $0.40\% \pm 0.003\%$, and an inference time of $0.83 \pm 0.02$ ms per sample. As reported in Table 4, all operational constraints are satisfied, generator headroom limits are respected, and the reserve requirement is fully met. Figure 5 further shows that the mean validation optimality gap decreases from approximately $0.521\%$ at epoch 1 to $0.427\%$ at epoch 30.

Given the feasible energy dispatch, we then apply the proportional reserve-allocation method in Section 3.2. As shown in Figure 4, the resulting reserve allocation satisfies the corresponding capacity and ramping constraints. This confirms that reserve feasibility can be recovered analytically, without learning, provided that the commitment offers sufficient reserve capability, i.e., $D_t + R_t \leq \sum_g P^{\max}_g u_{g,t}$.

## D.2 Sensitivity Study on Precision and Bisection Iterations

Figure 6 compares the maximum constraint violation, mean absolute optimality gap, and runtime for `float32` and `float64` under different numbers of bisection iterations. The `float32` implementation reaches a numerical floor

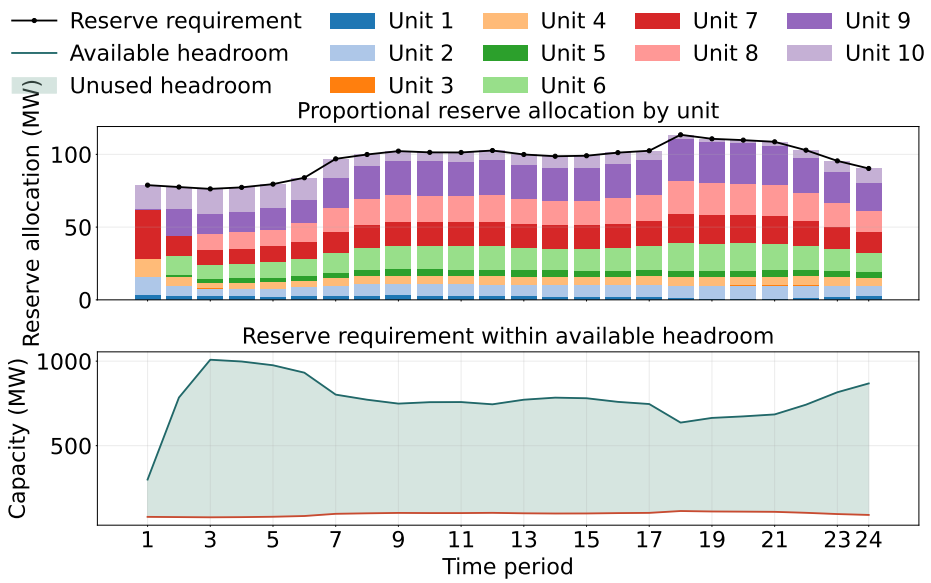


Figure 4: Allocation of reserve

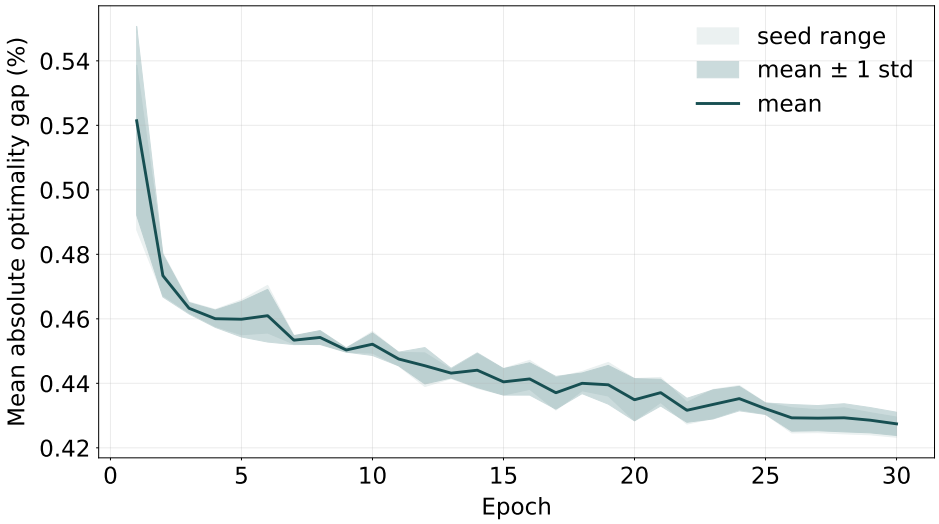


Figure 5: Training loss with the nonzero reserve datasets

of about $2.44 \times 10^{-4}$ MW, whereas `float64` reduces the residual to approximately $4.55 \times 10^{-13}$ MW. Increasing the number of bisection iterations improves numerical accuracy up to about 20 iterations, after which the improvement becomes negligible. We therefore recommend `float64` with 20–60 bisection iterations to achieve high numerical precision with modest additional computational cost.

### D.3 Performance with Various Configuration

We evaluate eight SMOP configurations to quantify their effects on feasibility and optimality. All configurations use the same dataset, GRU architecture, and label-free training strategy. As shown in Figure 7 and Table 5, removing backward propagation or sequential recovery substantially degrades feasibility, causing a 30.18 MW ramping violation and a 4.14 MW power-balance violation, respectively; removing backward propagation also increases the optimality gap by about 0.24%. The two loss-related variants both maintain zero violation, while the cost-focused weighting

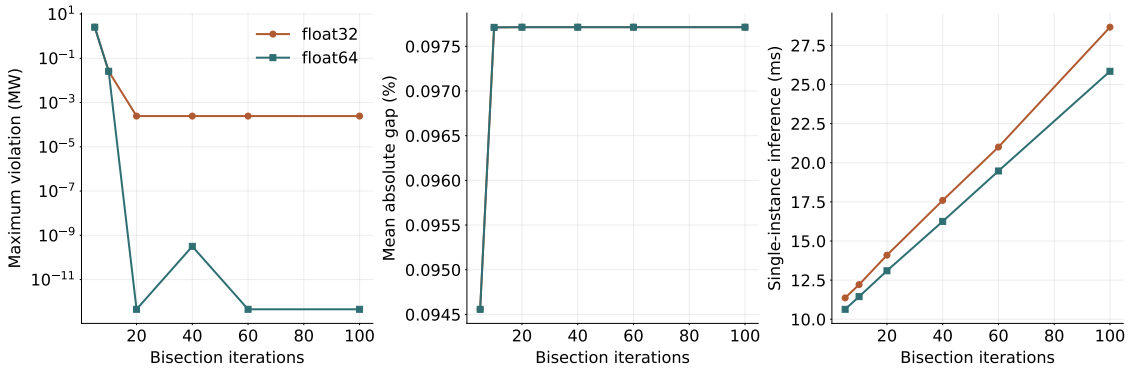


Figure 6: Sensitivity to precision and bisection iteration

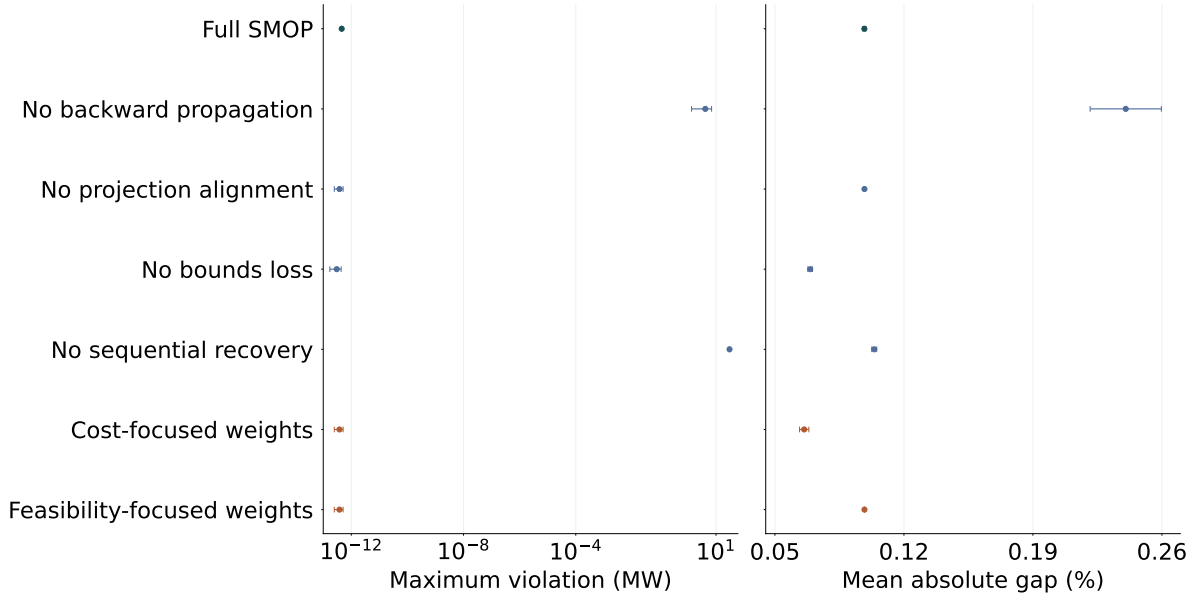


Figure 7: Sensitivity of Different configurations to feasibility and optimality

Table 5: SMOP ablations over three random seeds (mean ± standard deviation).

| Configuration | Max. violation (MW) | Abs. gap (%) |
|---|---|---|
| Full SMOP | $4.55 \times 10^{-13} \pm 0$ | $0.099 \pm 7.79 \times 10^{-4}$ |
| No backward propagation | $4.138 \pm 2.796$ | $0.240 \pm 0.019$ |
| No projection alignment | $3.79 \times 10^{-13} \pm 1.31 \times 10^{-13}$ | $0.099 \pm 8.19 \times 10^{-6}$ |
| No bounds loss | $3.03 \times 10^{-13} \pm 1.31 \times 10^{-13}$ | $0.069 \pm 1.27 \times 10^{-3}$ |
| No sequential recovery | $30.181 \pm 0.371$ | $0.104 \pm 1.33 \times 10^{-3}$ |
| Cost-focused weights | $3.79 \times 10^{-13} \pm 1.31 \times 10^{-13}$ | $0.066 \pm 2.53 \times 10^{-3}$ |
| Feasibility-focused weights | $3.79 \times 10^{-13} \pm 1.31 \times 10^{-13}$ | $0.099 \pm 8.80 \times 10^{-6}$ |

achieves the smallest optimality gap of $0.0659\% \pm 0.0025\%$. Overall, backward–forward propagation and sequential recovery are the key structural components for preserving SMOP feasibility.

Table 6: Interpretation of the SMOP ablation configurations.

| Configuration | Modification | Interpretation and result |
|---|---|---|
| Full SMOP | Uses forward and backward interval propagation, sequential recovery, box-balance projection, and all default self-supervised losses. | Serves as the complete reference method. |
| No backward propagation | Removes the backward pass that tightens earlier dispatch intervals using future ramp requirements. | An early dispatch may make future demand unreachable. The resulting $4.138$ MW maximum violation demonstrates the importance of future-aware propagation. |
| No projection alignment | Sets the projection-alignment loss weight to zero while retaining the hard projection. | The raw prediction is no longer encouraged to resemble the projected dispatch. Feasibility remains nearly exact, and the optimality gap changes only slightly. |
| No bounds loss | Sets all bounds-feasibility loss weights to zero. | Demand outside the aggregate propagated interval is no longer penalized directly. Hard recovery remains feasible on these samples. The smaller gap may reflect less competition with the cost objective, but does not show that the loss is unnecessary on harder instances. |
| No sequential recovery | Recovers periods independently instead of constructing period $t$'s interval from the recovered dispatch at $t-1$. | Individual periods may satisfy balance and capacity constraints while consecutive periods violate ramp constraints. This produces a $30.181$ MW maximum violation. |
| Cost-focused weights | Changes $\lambda_{\text{cost}} : 1 \to 2$, $\lambda_{\text{align}} : 1 \to 0.5$, $\lambda_{\text{raw}} : 0.1 \to 0.05$, and $\lambda^{\max}_{\text{bounds}} : 0.5 \to 0.25$. | Preserves all recovery mechanisms and obtains the smallest mean absolute gap, $0.0659\%$, while retaining numerical feasibility. |
| Feasibility-focused weights | Changes $\lambda_{\text{cost}} : 1 \to 0.5$, $\lambda_{\text{align}} : 1 \to 2$, $\lambda_{\text{raw}} : 0.1 \to 0.5$, and $\lambda^{\max}_{\text{bounds}} : 0.5 \to 1$. | Provides little additional benefit because hard recovery already produces essentially feasible solutions, while placing less emphasis on cost minimization. |

## E Reproducibility statement

We will release the code, configurations, and datasets needed to reproduce the results, including random seeds, hyperparameters, and solver settings. Hardware specifications are reported for reference, but the experiments can be reproduced on standard CPU/GPU platforms.

## F The Use of Large Language Models (LLMs)

We use LLMs for grammar and wording refinement, while independently verifying all technical content for accuracy and reliability.