Proceedings of Machine Learning Research vol vvv:1–18, 2025

A Dynamic Penalization Framework for Online Rank-1 Semidefinite Programming Relaxations

Ahmad Al-Tawaha ATAWAHA@VT.EDU
Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

Javad Lavaei LAVAEI@BERKELEY.EDU
Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, USA

Ming Jin JINMING@VT.EDU
Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

Abstract

We propose a dynamic penalization framework for recovering rank-1 solutions in sequential semidefinite programming (SDP) relaxations. Obtaining rank-1 solutions—crucial for recovering physically meaningful solutions in many applications—becomes particularly challenging in dynamic environments where problem parameters continuously evolve. Our framework operates in two interconnected phases: the learning phase dynamically adjusts penalty parameters to enforce rank-1 feasibility based on feedback from the decision phase, while the decision phase solves the resulting penalized SDP relaxations using the penalty parameters specified by the learning phase. To accelerate rank-1 recovery across sequential problems, we introduce a meta-learning model that provides informed initializations for the penalty matrices. The meta-learning model leverages historical data from previously solved tasks, eliminating the need for externally curated datasets. By using task-specific features and updates from prior iterations, the meta-model intelligently initializes penalty parameters, reducing the number of iterations required between the two phases. We prove sublinear convergence to rank-1 solutions and establish low dynamic regret bounds that improve with task similarity. Empirical results on real-world rank-constrained applications, including the Max-Cut problem and Optimal Power Flow (OPF), demonstrate that our method consistently recovers rank-1 solutions.

Keywords: Online Optimization, Semidefinite Programming, Dynamic Penalization, Meta-Learning, Rank Constraint.

1. Introduction

Semidefinite programming has emerged as a powerful framework for addressing nonlinear and nonconvex optimization problems, particularly in polynomial optimization and quadratically constrained quadratic programs (QCQPs). A wide range of optimization problems, including discrete optimization problems, can be reformulated or approximated as polynomial optimization problems, which are further transformable into QCQPs using auxiliary variables and constraints (Madani et al., 2020; Wang, 2022). By relaxing the nonconvex QCQP constraints into semidefinite problem, SDPs enable exact solutions or tight lower bounds on the objective (Boyd et al., 1994; Boyd and Vandenberghe, 2004). However, enforcing rank-1 constraints in these relaxations remains challenging. Convex SDP relaxations yield low-rank solutions efficiently, but achieving exact rank-1 solutions often requires penalty terms that drive nonzero eigenvalues to zero, promoting rank-1 feasibility (Pataki, 1998; Sojoudi and Lavaei, 2014).

This challenge is further compounded in sequential optimization settings, where evolving problem parameters require solving a series of similar instances, as seen in power systems (Dall'Anese et al., 2017), communication networks (Chen and Lau, 2011), and online learning (Mokhtari et al., 2016). Solving each instance independently is computationally prohibitive, even with modern advances (Zavala and Anitescu, 2010). Traditional SDP-based methods with pre-tuned penalty parameters for static problems (Zohrizadeh et al., 2018; Madani et al., 2015; Liu et al., 2017) fail in online tasks, where re-optimizing penalty terms for each new instance adds significant overhead.

A promising direction to mitigate these computational challenges further lies in meta-learning techniques (Hospedales et al., 2021), which accelerate optimization by leveraging knowledge from previously solved similar problems. This approach could enable faster convergence through intelligent initialization of key parameters, such as the penalty term parameter in penalized SDPs. However, applying meta-learning to sequential rank-1 constrained SDPs requires addressing unique theoretical and practical challenges, particularly in guaranteeing rank-1 recovery while maintaining computational efficiency.

The fundamental challenge in sequential relaxed SDPs is achieving computationally efficient rank-1 recovery in dynamic environments. Determining penalty terms that consistently enforce rank-1 solutions is inherently complex (Mezura-Montes and Coello, 2011). Although static (Hsieh et al., 2015), dynamic (Liu et al., 2016), and adaptive penalty methods (Wang et al., 2021; Krohling and dos Santos Coelho, 2006; Fan and Yan, 2012; Huang et al., 2007; Wang et al., 2023) exist for constrained optimization, most rely on stochastic adjustments and are designed for offline scenarios. Moreover, traditional SDP relaxations use pre-tuned penalty terms for static problems, which become infeasible in evolving settings, as they require re-optimization for every new instance. These inefficiencies highlight the need for adaptive and automated frameworks that dynamically adjust penalty parameters within single and across sequential tasks.

First-order methods, including augmented Lagrangian (Wang and Hu, 2023), accelerated gradient (Wang and Kılınç-Karzan, 2024), and conditional gradient algorithms, have significantly improved SDP scalability. While these methods, along with penalization approaches for specific combinatorial problems (Wang et al., 2019; Krechetov et al., 2019), excel at solving low-rank or structured SDPs, they often require careful problem-specific designs. Our work complements these advances by proposing a meta-learning framework that automatically adapts penalty matrices across multiple tasks. Compatible with any differentiable SDP solver, our framework generalizes to diverse problems, ranging from simple combinatorial optimization to complex AC-OPF instances.

Two key challenges arise: dynamically selecting penalty parameters for rank-1 constraints in evolving problems, and exploiting similarities across sequential instances for efficient optimization. We address these by proposing a framework that combines dynamic penalization with meta-learning for solving sequences of relaxed SDPs. Our approach adaptively tunes penalties per task while using meta-learning for intelligent parameter initialization, reducing computational costs. The main contributions of this work are as follows.

Dynamic Penalization for Rank-1 Recovery: We develop a dynamic penalization framework where $W_t \in \mathbb{S}^n$ represents the penalty parameter dynamically adjusted for each task t. This adjustment is based on differentiation through the relaxed SDP problem, ensuring efficient enforcement of the rank-1 constraint while maintaining computational feasibility. The framework leverages established convergence guarantees, demonstrating a sublinear convergence rate (Lemma 1).

Meta-Initialization for Multi-Task SDP Optimization: To accelerate convergence across tasks, we introduce a meta-learning approach that initializes the penalty parameter $W_{t,1}$ for each task t using task-specific features ϕ_t and historical data. This reduces the number of iterations required K_t for convergence, especially when tasks exhibit similarities.

Dynamic Regret Analysis: Leveraging dynamic regret, we derive a bound on the average iterations K_t required per task, showing that it scales as $\mathcal{O}\left(T^{-1/6}\right)$ with the number of tasks T. The bound also depends on the path length V_T, which quantifies the rate of change between tasks, and the relatedness to the task $\mathcal{S}_{W^*}$, highlighting the framework's ability to adapt to task similarity for improved efficiency (Lemma 2).

The organization of the paper is as follows: Sec. 2 introduces the problem formulation, Secs. 3 and 4 describe our method for single-task and multi-task scenarios, respectively, Sec. 5 presents the experimental evaluations, and Sec. 6 concludes the paper. Due to space limitations, all proofs and additional experimental details are provided in the online supplement (Al-Tawaha et al., 2024).

2. Problem Formulation

We consider an online sequence of polynomial optimization tasks $\{\mathcal{T}_t\}_{t=1}^T$ that arrive sequentially. Each instance $\mathcal{T}_t$ represents varying operational conditions, commonly encountered in real-world applications. To model this, we express each task as a QCQP:

$$\underset{x_t\in\mathbb{R}^n}{\text{minimize}}; x_t^\top M_0^t x_t \quad \text{subject to} \quad x_t^\top M_i^t x_t \le a_i^t, ; i=1,\ldots,p, \quad x_t^\top N_j^t x_t = b_j^t, ; j=1,\ldots,q. \tag{1}$$

where $M_0^t, M_i^t, N_j^t \in \mathbb{S}^n$ are symmetric matrices representing the coefficients of the polynomial functions for task $\mathcal{T}_t$, and $a_i^t, b_j^t \in \mathbb{R}$ are scalars associated with the inequality and equality constraints, respectively. To tackle the non-convex QCQP (1), we introduce an auxiliary variable $X_t = x_t x_t^\top \in \mathbb{S}^n$. This allows us to reformulate the problem as a rank-constrained SDP:

$$\begin{aligned}
\underset{X_t\in\mathbb{S}^n}{\text{minimize}} \quad & f_0^t(X_t) && \text{(2a)}\\
\text{subject to} \quad & f_i^t(X_t) \le a_i^t, \quad i=1,\ldots,p, && \text{(2b)}\\
& h_j^t(X_t) = b_j^t, \quad j=1,\ldots,q, && \text{(2c)}\\
& X_t \succeq 0, && \text{(2d)}\\
& \operatorname{rank}(X_t) = 1, && \text{(2e)}
\end{aligned}$$

where $f_i^t, h_j^t : \mathbb{S}^n \to \mathbb{R}$ are twice continuously differentiable convex functions for $i = 0,\ldots,p$ and $j = 1,\ldots,q$. While the reformulation captures the original problem, the rank-one constraint (2e) is non-convex, making the problem computationally challenging. To address this, we relax the rank-one constraint and introduce a penalty term $g\left(X_t; W_t\right)$, a μ-strongly convex function of X_t, to encourage low-rank solutions. Here, $W_t \in \mathbb{S}^n$ represents designable parameters associated with the penalty. The resulting penalized semidefinite SDP is formulated as:

$$\underset{X_t\in\mathbb{S}^n}{\text{minimize}}\ f_0^t(X_t) + g(X_t; W_t) \quad \text{subject to} \quad \begin{aligned} & f_i^t(X_t) \le a_i^t,\ i=1,\ldots,p,\\ & h_j^t(X_t) = b_j^t,\ j=1,\ldots,q,\\ & X_t \succeq 0. \end{aligned} \tag{3}$$

The penalty function $g : \mathbb{S}^n \times \mathbb{S}^n \to \mathbb{R}$ is defined to guide X_t toward being approximately $\operatorname{rank}-1$ while maintaining the convexity of the problem. For large-scale instances, enforcing full positive semidefinite constraints on X_t is computationally prohibitive. To address this, we follow the approach in Madani et al. (2015) and impose positive semidefinite constraints on principal submatrices of X_t corresponding to selected subgraphs. For each problem instance $\mathcal{T}_t$, a cycle basis $\mathcal{C}_1^t, \ldots, \mathcal{C}_q^t$ in the graph $\mathcal{G}^t$, along with edges not included in these cycles, forms the set of subgraphs Ω^t. The reduced SDP and penalized reduced SDP relaxation are formulated as:

$$\underset{X_t \in \mathbb{S}^n}{\text{minimize}} \; f_0^t(X_t) + g(X_t; W_t) \quad \text{subject to} \quad \begin{aligned} & f_i^t(X_t) \leq a_i^t, \; i = 1, \ldots, p, \\ & h_j^t(X_t) = b_j^t, \; j = 1, \ldots, q, \\ & X_t\{\mathcal{G}_s^t\} \succeq 0, \; \forall \mathcal{G}_s^t \in \Omega^t, \end{aligned} \tag{4}$$

where $X_t \left\{\mathcal{G}_s^t\right\}$ denotes the principal submatrix of X_t corresponding to the nodes in $\mathcal{G}_s^t$. By restricting positive semidefinite constraints to these submatrices, the computational complexity is reduced while preserving essential structural properties. The rank-1 constraint is imposed on the submatrices as $\operatorname{rank}\left(X_t\left\{\mathcal{G}_s^t\right\}\right) = 1$ for all $\mathcal{G}_s^t \in \Omega^t$. This ensures the rank of the reconstructed matrix X_t is guaranteed to satisfy $\operatorname{rank}(X_t) \leq \max\left\{\operatorname{rank}\left(X_t\left\{\mathcal{G}_s^t\right\}\right) \mid \mathcal{G}_s^t \in \Omega^t\right\}$, and since the rank of each submatrix is constrained to one , it follows that $\operatorname{rank}(X_t) = 1$. Note that there is a relationship between (2), (3), and (4). Let us define f^*, f^*_{SDP}, $f^*_{\text{r}-\text{SDP}}$, and $f^*_{\varepsilon,\text{r}-\text{SDP}}$, and $f^*_{\varepsilon,\text{SDP}}$ as the optimal solutions of the original problem (2), the standard SDP relaxation “problem (3) without penalty term”, the reduced SDP relaxation “problem (4) without penalty term, the penalized reduced SDP relaxation (4) with rank-1 (feasible) solution, , and the penalized SDP relaxation (3) with rank-1 (feasible) solution, respectively. By comparing the feasible sets of these optimization problems (Sojoudi and Lavaei, 2014; Madani et al., 2015), we have: $f^*_{\text{r}-\text{SDP}} \leq f^*_{\text{SDP}} \leq f^* \leq f^*_{\varepsilon,\text{SDP}} \leq f^*_{\varepsilon,\text{r}-\text{SDP}}$ the significance of this relationship is that it allows us to quantify how close our obtained solution is to the global optimum. Specifically, we can define a **global optimality guarantee** as:

$$G_{\text{opt}} = 100 - \frac{f^*_{\varepsilon,\text{r}-\text{SDP}} - f^*_{\text{r}-\text{SDP}}}{f^*_{\varepsilon,\text{r}-\text{SDP}}} \times 100\%. \tag{5}$$

3. Dynamic Penalization for Single-Task Rank-1 Enforcement in SDP

For each task $\mathcal{T}_t$ in the online sequence $\{\mathcal{T}_t\}_{t=1}^T$, our framework operates in two interconnected phases: a learning phase and a decision phase. In the learning phase, the objective is to iteratively refine the penalty parameters W_t, ensuring efficient rank-1 feasibility enforcement. This phase leverages a loss function $\mathcal{L}(X_t, W_t)$ that penalizes deviations from the rank-1 property, guiding X_t toward being approximately rank-1. The optimization problem for updating W_t is formulated as:

$$\underset{W \in \mathbb{S}^n}{\text{minimize}} \; G(W_t) = \mathcal{L}(X_t^*(W_t), W_t), \tag{6}$$

where X_t^* is the optimal solution function obtained in the decision phase. This phase computes X_t^* by solving the following penalized SDP problem:

$$X_t^*(W_t) = \underset{\bar{X}_t \in \mathbb{S}^n}{\arg\min} \; f_0^t(\bar{X}_t) + g(\bar{X}_t; W_t) \quad \text{subject to} \quad \begin{aligned} & f_i^t(\bar{X}_t) \leq a_i^t, \quad i = 1, \ldots, p, \\ & h_j^t(\bar{X}_t) = b_j^t, \quad j = 1, \ldots, q, \\ & \bar{X}_t \succeq 0, \end{aligned} \tag{7}$$

we assume that $X_t^*(W_t)$ exists and satisfies the Linear Independence Constraint Qualification (LICQ) at the optimal point. For instance, this condition holds in optimal power flow problem (Hauswirth et al., 2018), which is the focus of our experimental validation.

We aim to iteratively refine W_t to guide X_t toward a rank-1 solution by optimizing the loss function $G(W_t) = \mathcal{L}(X^*(W_t), W_t)$ with respect to W_t. Importantly, $G(W_t)$ depends on the optimal solution $X_t^*(W_t)$ obtained during the decision phase, which solves a constrained optimization problem. Gradient-based updates to W_t involve differentiating $G(W_t)$, requiring the computation of $\frac{\partial X_t(W_t)}{\partial W_t}$. However, this derivative is often challenging to compute. Even if the learning and decision phases employ convex relaxations and differentiable functions (Liu et al., 2021), $G(W_t)$ remains generally nonconvex and non-differentiable due to the constraints in the decision phase optimization, which can render the mapping $W_t \mapsto X_t^*(W_t)$ non-smooth. To overcome these challenges, our framework integrates subgradient methods, effectively managing cases where $G(W_t)$ is subdifferentiable or admits a generalized gradient.

At iteration k for task t, the penalty parameter is denoted as $W_{t,k}$, and the descent direction is represented as $g_{t,k}$. When $G(W_t)$ is differentiable in a neighborhood of $W_{t,k}$, the descent direction is given by the gradient $g_{t,k} = \nabla G(W_{t,k})$. In cases where $G(W_t)$ is non-differentiable, $g_{t,k}$ is defined following the approach in Xu and Zhu (2023) as the vector with the smallest norm in the convex hull of the generalized gradient set $\partial G(W_{t,k}, \varepsilon_k)$, where ε_k is a tolerance parameter. This approach ensures a valid descent direction irrespective of the differentiability of $G(W_t)$.

We iteratively update W_t over K_t iterations, computing the sequence $\{W_{t,k}, X_{t,k}\}_{k=0}^{K}$. The objective $G(W_t)$ is optimized using a line search procedure to guarantee sufficient decrease at each step. This iterative scheme progressively refines $W_{t,k}$, guiding $X_t^*(W_t)$ toward satisfying the rank-1 constraint while solving the convex relaxation of the original problem. Furthermore, the iterative process ensures that the rank-1 loss decays sublinearly with K, as shown in the following

Lemma 1 *Let $G(W_t)$ be the penalized objective function and $g_{t,k}$ the descent direction at iteration k for task t. Assume $G(W_t)$ is bounded below by G_{inf} and satisfies the sufficient decrease condition. Then, after K iterations, the average gradient norm is bounded as:*

$$\frac{1}{K}\sum_{k=1}^{K} \|g_{t,k}\| \leq \sqrt{\frac{G(W_{t,1}) - G(W_{t,K})}{\beta \sigma_{min} K}}$$

where $\beta \in (0,1)$ is the line search parameter and $\sigma_{min} > 0$ is a lower bound on the step size.

This result guarantees sublinear convergence of the gradient norm in $O(1/\sqrt{K})$, ensuring efficient optimization of the penalized objective.

4. Meta-Learning for Multi-Task Rank-1 Enforcement in SDP

Solving SDP problems with rank-one constraints can be computationally intensive, especially in multi-task settings where each task may require many iterations to converge. To address this challenge, we propose a meta-learning approach that leverages information from previous tasks to accelerate optimization for new tasks. By capturing shared structures among tasks, we can improve efficiency for solving new problems. The meta-learning framework operates in conjunction with the learning phase, providing a strong starting point for optimization by predicting effective initializations $W_{t,1}$. This reduces the number of iterations required for interaction between the learning and

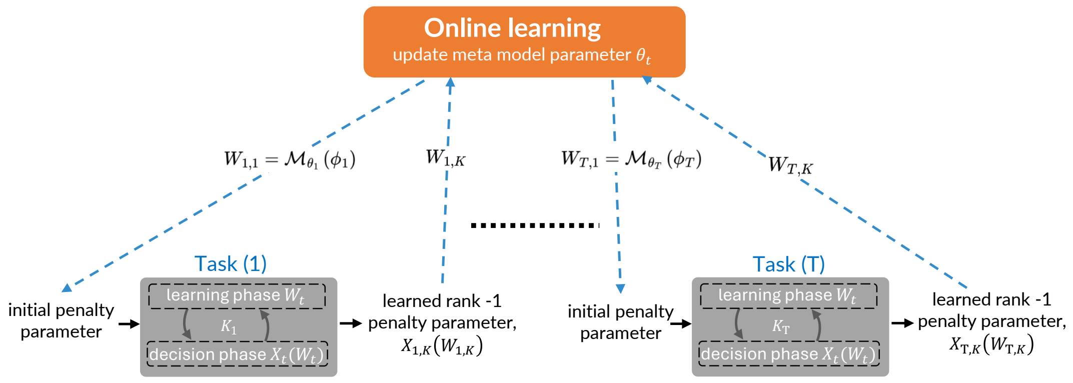
Online learning
update meta model parameter θ_t
$W_{1,1} = \mathcal{M}_{\theta_1}(\phi_1)$
$W_{1,K}$
$W_{T,1} = \mathcal{M}_{\theta_T}(\phi_T)$
$W_{T,K}$
Task (1)
initial penalty parameter
learning phase W_t
K_1
decision phase $X_t(W_t)$
learned rank -1 penalty parameter, $X_{1,K}(W_{1,K})$
Task (T)
initial penalty parameter
learning phase W_t
K_T
decision phase $X_t(W_t)$
learned rank -1 penalty parameter, $X_{T,K}(W_{T,K})$

Figure 1: Meta-learning framework for multi-task rank-one enforcement in SDP. The upper-level optimization updates the penalty matrix W_t to minimize $G(W_t)$, while the lower-level solves the constrained SDP to compute $X_t(W_t)$. The meta-model $\mathcal{M}_\theta$ predicts an effective initialization $Wt, 1$ using features ϕ_t, accelerating convergence and progressively enforcing the rank-one constraint.

decision phases. We train a meta-model $\mathcal{M}_\theta$, parameterized by θ_t, to predict effective initializations $W_{t,1}$ for the penalty matrix based on task-specific features ϕ_t. This provides a strong starting point for the optimization algorithm, reducing the number of iterations K_t needed for convergence (see Figure (1)).

For each task t, we extract features ϕ_t that encapsulate critical characteristics. The meta-model predicts the initial penalty matrix $W_{t,1} = \mathcal{M}_{\theta_t}(\phi_t)$. Starting from $W_{t,1}$, we solve the penalized SDP problem using an iterative scheme, refining $W_{t,1}$ to $W_{t,K}$ after K iterations (detailed in Section 3). We evaluate the meta-model's performance using the meta-loss function:

$$\ell_t(\theta_t) = \|\mathcal{M}_{\theta_t}(\phi_t) - W_{t,K}\|^2 .$$

The meta-model parameters θ_t are updated via an online learning algorithm to minimize the metaloss, resulting in parameters θ_{t+1} for the next task. Specifically, we use Online Gradient Descent (OGD) for convex loss functions (Besbes et al., 2015; Al-Tawaha and Jin, 2024) and Follow-the-Perturbed-Leader (FTPL-A) for non-convex loss functions (Xu and Zhang, 2024). This iterative update mechanism effectively transfers knowledge across tasks, reducing computational overhead while maintaining high optimization quality. Algorithm (1) formalizes this framework.

Algorithm 1 Meta-Learning Framework for Initialization

```
Require: Number of problem instances $T$, initial meta-model parameters $\theta_0$
1: for $t = 1$ to $T$ do
2:     Receive problem instance $\mathcal{T}_t$
3:     Extract problem-specific features $\phi_t$
4:     Initialize penalty matrix using the meta-model: $W_{t,1} = \mathcal{M}_{\theta_t}(\phi_t)$
5:     Solve problem (6) starting from $W_{t,1}$ using the method from Xu and Zhu (2023), and obtain
       $W_{t,K_t}, X_{t,K_t}$ after $K_t$ iterations
6:     Compute the meta-loss: $\ell_t(\theta_t) = \|\mathcal{M}_{\theta_t}(\phi_t) - W_{t,K}\|^2$
7:     Update the meta-model parameters $\theta_{t+1}$ using an online learning algorithm to minimize $\ell_t(\theta_t)$
8: end for
```

Our meta-learning framework accelerates convergence and reduces computational costs by leveraging shared structures across tasks and adapting initialization predictions dynamically. We now analyze its performance through dynamic regret bounds and average iteration guarantees. Analyzing the dynamic regret of the optimization process provides insight into the efficiency and adaptability of our meta-learning framework across multiple tasks. The dynamic regret quantifies the cumulative difference between the performance of our online algorithm and that of the best possible sequence of decisions in hindsight: $R_T^{\mathrm{d}} = \sum_{t=1}^{T} \ell_t(\theta_t) - \sum_{t=1}^{T} \ell_t(\theta_t^*)$, where θ_t^* denotes the optimal parameters for task t. By bounding the dynamic regret, we derive performance guarantees that relate to the expected average number of iterations required per task. The following lemma encapsulates our main theoretical results.

Lemma 2 *Consider a sequence of T optimization tasks $\{\mathcal{T}_t\}_{t=1}^T$ with a predictive model $\mathcal{M}_{\theta_t}$ parameterized by $\theta_t \in \Theta$ for initialization. For both convex and non-convex meta-loss functions, when θ_t is updated using OGD (convex case) or follow-the-Perturbed-Leader (FTPL) variant in Xu and Zhang (2024) (non-convex case), the expected average number of iterations required per task to achieve gradient norm less than δ is bounded by:*

$$\frac{1}{T}\sum_{t=1}^{T} K_t \leq \frac{L_{max}}{\beta\sigma_{min}\delta^2}\left(\sqrt{\max\{\mathcal{C}_1,\mathcal{C}_2\}}V_T^{1/6}T^{-1/6} + \mathcal{S}_{W^*}\right) \tag{8}$$

where $V_T = \sum_{t=2}^{T} \left\|\theta_t^* - \theta_{t-1}^*\right\|$ is the path length, $\mathcal{S}_{W^*}^2 = \frac{1}{T}\sum_{t=1}^{T} \left\|\mathcal{M}_{\theta_t^*}(\phi_t) - W_t^*\right\|^2$ is the task-relatedness with respect to a sequence of changing comparator optimal solution $\{W_t^*\}_{t=1}^T$and $\mathcal{C}_1$ and $\mathcal{C}_2$ are constants that depends on the diameter of parameter space $D = \max_{\theta_1,\theta_2\subset\Theta} \|\theta_1 - \theta_2\|_\infty$, and a bound on the gradient norms $\|\nabla\ell_t(\theta)\| \leq \ell, \forall\theta_t \in \Theta, \forall t$, and $L_{\max} = \max_{1\leq t\leq T}\max_{1\leq k\leq K_t} L_{G,t,k}$ is based on the local Lipschitz continuity at each iteration. Specifically, for each iteration k in task t, the algorithm generates $W_{t,k}$. In a neighborhood around $W_{t,k'}$, $G_t(W)$ is Lipschitz continuous with Lipschitz constant $L_{G,t,k}$.

Our analysis reveals two principal insights regarding the efficiency and adaptability of the proposed framework. First, as the number of tasks T increases, the average number of iterations required per task decreases proportionally to $T^{-1/6}$. This demonstrates that the meta-model becomes increasingly effective at predicting suitable initializations as it learns from prior tasks, thereby reducing the computational effort required for new tasks. Importantly, the sublinearity of our approach holds as long as the path length V_T grows sublinearly with T, a standard assumption in dynamic regret analyses. The path length $V_T = \sum_{t=2}^{T} \left\|\theta_t^* - \theta_{t-1}^*\right\|$ quantifies the cumulative change in the optimal parameters between successive tasks. A smaller V_T suggests that the optimal parameters vary minimally between tasks, indicating higher task similarity.

Second, task similarity plays a crucial role in optimization efficiency, as captured by both the path length V_T and the task-relatedness measure $\mathcal{S}_{W^*}$. The measure $\mathcal{S}_{W^*}^2 = \frac{1}{T}\sum_{t=1}^{T} \|\mathcal{M}_{\theta_t^*}(\phi_t) - W_t^*\|^2$ represents the average discrepancy between the meta-model's predicted initializations and the true optimal penalty matrices across tasks. When tasks are similar, their optimal penalty matrices W_t^* are also similar. The meta-model effectively captures and generalizes this shared structure using task features ϕ_t, allowing it to predict initializations that are consistently closer to W_t^*. This leads to fewer iterations required for convergence, as the optimization begins closer to the optimal solution for each task.

5. Experiments

5.1. Dynamic penalization for single-task SDP: a Max-Cut case study

The Max-Cut problem seeks to partition the vertices of a graph $G = (V, E)$ into two sets such that the sum of the weights of edges between the sets is maximized. The standard SDP relaxation of this problem replaces the binary constraints $x_i \in \{-1, 1\}$ with a semidefinite constraint on the Gram matrix X :

$$\begin{array}{ll} \max_{X \in \mathbb{S}^n} & \frac{1}{4}\operatorname{Tr}(LX) \\ \text{s.t.} & X \succeq 0, \quad X_{ii} = 1, \forall i \in V, \end{array} \tag{9}$$

where L is the Laplacian matrix of the graph $G = (V, E)$, defined as $L = D - C$, with D being the degree matrix and C the adjacency matrix containing the edge weights c_{ij}. Interestingly, the Max-Cut SDP formulation demonstrates differentiable behavior in the mapping $X(W)$, enabling the use of `cvxpylayer` to compute gradients. Our framework iteratively refines penalty parameters W to enforce rank-1 feasibility. Each iteration consists of a learning phase, which updates W to minimize:

$$G(W) = \lambda_1 \left(n^2 - \|X\|_F^2\right) + \lambda_2 \operatorname{Tr}(LX),$$

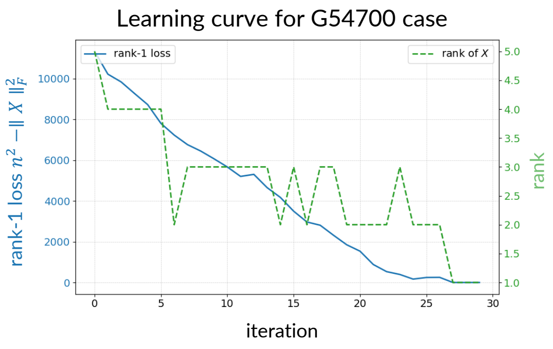

Figure 2: Learning curve for G54700 illustrating the effective rank of the solution across iterations.

and a decision phase, where $X$is computed by solving the penalized SDP:

$$X \in \arg\max_{\bar{X} \in \mathbb{S}^n} \quad \frac{1}{4}\operatorname{Tr}(L\bar{X}) - \left(\operatorname{Tr}(W\bar{X}) + \mu\|\bar{X}\|_F^2\right), \text{ s.t.} \qquad \bar{X} \succeq 0, \quad \bar{X}ii = 1, ; \forall i \in V.$$

The process is repeated for K iterations, with feedback from the decision phase guiding updates in the learning phase. Our method was tested on 10 small graphs from Set 2 of the Optsicom Project benchmark [1], each containing $n = 125$ nodes and $m = 375$ edges. The results shown in Table (1) demonstrate that our approach performs well, often matching the optimal solution or the existing methods. Using the G54700 graph as an example, Figure (2) tracks how both the rank-1 loss ($n^2 - \|X\|_F^2$) and effective rank improve over iterations. Unlike traditional SDP methods, our approach naturally produces rank-1 solutions without randomization - a key advantage for differentiable optimization as it enables end-to-end gradient flow without stochastic rounding, making it ideal for deep learning pipelines.

5.2. Dynamic penalization for multi-task SDP: Optimal Power Flow case study

As a practical case study, we apply the proposed approach to the OPF problem for multi-task online setting, where each task t corresponds to a unique operational condition. The OPF problem seeks to minimize generation costs while satisfying operational constraints such as power balance, voltage limits, and line flow capacities. To address scalability, we use the reduced SDP relaxation in (4),

1. http://grafo.etsii.urjc.es/optsicom/maxcut/

Table 1: Comparison of Results Across Graphs and Methods. RUN-CSP results are from Toenshoff et al. (2021), and Khalil et al. results are from Khalil et al. (2017).

Graphs	Dyn. Penalized SDP	RUN-CSP	Khalil et al.	Greedy Search	Opt
G54100	**110**	**110**	108	80	110
G54200	108	**112**	108	90	112
G54300	**106**	**106**	104	86	106
G54400	**112**	**112**	108	96	114
G54500	110	**112**	**112**	94	112
G54600	**110**	**110**	**110**	88	110
G54700	**112**	110	108	88	112
G54800	**108**	106	**108**	76	108
G54900	**108**	**108**	**108**	88	110
G541000	**110**	**110**	108	80	112

where rank-1 feasibility is achieved when all submatrices corresponding to selected subgraphs have rank-1. For each task t, the penalized reduced SDP formulation as:

$$\min_{X_t \in \mathbb{S}^n} \sum_{i \in \mathcal{G}} c_i^t \left(P_{G_i}^t \right) + \operatorname{Tr}\left(W_t X_t \right),$$

where c_i^t and $P_{G_i}^t$ are the generation cost and active power of generator i, respectively, and the penalty term $\operatorname{Tr}(W_t X_t)$ enforces rank-1 solutions. The learning phase loss function, $\mathcal{L}(X_t^*, W_t) = \|X_t^*\|_* - \|X_t^*\|_2$, measures deviation from rank-1 feasibility. Our meta-learning model $\mathcal{M}_{\theta_t}$ initializes the penalty matrix $W_{t,1}$ based on task-specific features and prior tasks. Further details can be found in Al-Tawaha et al. (2024). Through iterative updates, $W_{t,1}$ evolves to $W_{t,K}$, ensuring feasibility and efficient rank-1 convergence. This process reduces computational overhead by learning from previous tasks.

Figure (3) presents the performance of the proposed dynamic penalization framework for solving sequential OPF problems across tasks with varying levels of similarity. The left-hand plots illustrate the number of iterations required to achieve rank-1 solutions, where all submatrices corresponding to selected subgraphs also satisfy the rank-1 condition, as the number of tasks increases. For highly similar tasks, the meta-model performance is similar to baseline methods, such as warm-start and moving-average initializations, which is expected due to the limited variation between tasks. However, as task similarity decreases, the advantages of the meta-model become more pronounced.

For less similar tasks, the meta-model significantly outperforms the baselines. The baseline methods often fail to converge, reaching the maximum allowable iterations without finding a feasible rank-1 solution. In contrast, the meta-model maintains robust performance, leveraging its ability to generalize across tasks and adapt to dynamic changes. This efficiency is reflected in the consistent reduction of iteration counts as the meta-model learns from an increasing number of tasks, aligning with our theoretical findings.

The right-hand plots show the global optimality gap, as defined in (5). Across all levels of task similarity, the proposed framework achieves a gap of $99.3 - 100\%$, underscoring its ability to maintain high-quality solutions that closely approximate the global optimum. Notably, even for less similar tasks where baselines diverge or fail to converge, the meta model preserves both feasibility and optimality, ensuring reliable solutions.

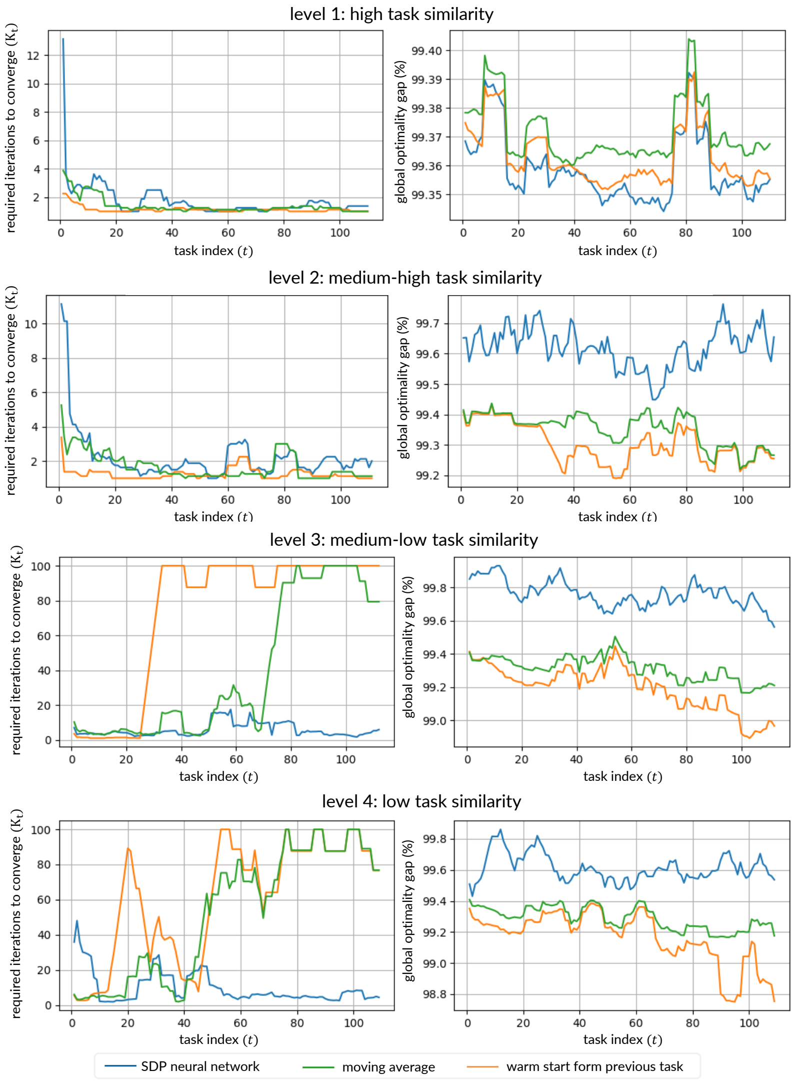

Figure 3: Comparison of initialization strategies across task similarity levels (level 1: high similarity to level 4: low similarity). Left: iterations to rank-1 convergence. Right: global optimality gap. Strategies include SDP neural network, moving average, and warm start from the previous task.

6. conclusion

In conclusion, we introduced a meta-learning framework that efficiently addresses rank-one constrained SDPs across multiple tasks by integrating dynamic penalization and meta-initialization strategies. Our dynamic penalization approach adaptively adjusts the penalty matrix W_t through differentiation of the relaxed SDP, ensuring effective rank-one enforcement with sublinear convergence guarantees. The meta-initialization leverages task-specific features and historical data to predict $W_{t,1}$, reducing the number of iterations K_t required for convergence, especially when tasks exhibit high similarity. Through dynamic regret analysis, we established that the average number of iterations per task scales as $\mathcal{O}\left(T^{-1/6}\right)$, contingent on the path length V_T and the task-relatedness measure $\mathcal{S}_{W^*}$, demonstrating the framework's adaptability to task similarity.

References

Ahmad Al-Tawaha and Ming Jin. Does online gradient descent (and variants) still work with biased gradient and variance? In *2024 American Control Conference (ACC)*, pages 3570–3575. IEEE, 2024.

Ahmad Al-Tawaha, Javad Lavaei, and Ming Jin. A dynamic penalization framework for online rank-1 semidefinite programming relaxations-online document. 2024. URL `https://ahmad-tawaha.webador.com/publication`.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations research*, 63(5):1227–1244, 2015.

Subhonmesh Bose, Dennice F Gayme, K Mani Chandy, and Steven H Low. Quadratically constrained quadratic programs on acyclic graphs with application to power flow. *IEEE Transactions on Control of Network Systems*, 2(3):278–287, 2015.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

Junting Chen and Vincent KN Lau. Convergence analysis of saddle point problems in time varying wireless systems—control theoretical approach. *IEEE Transactions on Signal Processing*, 60(1): 443–452, 2011.

Emiliano Dall'Anese, Swaroop S Guggilam, Andrea Simonetto, Yu Christine Chen, and Sairaj V Dhople. Optimal regulation of virtual power plants. *IEEE Transactions on Power Systems*, 33 (2):1868–1881, 2017.

Qinqin Fan and Xuefeng Yan. Differential evolution algorithm with co-evolution of control parameters and penalty factors for constrained optimization problems. *Asia-Pacific Journal of Chemical Engineering*, 7(2):227–235, 2012.

Adrian Hauswirth, Saverio Bolognani, Gabriela Hug, and Florian Dörfler. Generic existence of unique lagrange multipliers in ac optimal power flow. *IEEE control systems letters*, 2(4):791–796, 2018.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169, 2021.

Yi-Chih Hsieh, Yung-Cheng Lee, and Peng-Sheng You. Solving nonlinear constrained optimization problems: An immune evolutionary based two-phase approach. *Applied Mathematical Modelling*, 39(19):5759–5768, 2015.

Fu-zhuo Huang, Ling Wang, and Qie He. An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and computation*, 186(1):340–356, 2007.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Mikhail Krechetov, Jakub Marecek, Yury Maximov, and Martin Takac. Entropy-penalized semidefinite programming. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1123–1129. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/157. URL https://doi.org/10.24963/ijcai.2019/157.

Renato A Krohling and Leandro dos Santos Coelho. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(6):1407–1416, 2006.

Jianjun Liu, Kok Lay Teo, Xiangyu Wang, and Changzhi Wu. An exact penalty function-based differential search algorithm for constrained global optimization. *Soft Computing*, 20:1305–1313, 2016.

Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10045–10067, 2021.

Tian Liu, Bo Sun, and Danny HK Tsang. Rank-one solutions for sdp relaxation of qcqps in power systems. *IEEE Transactions on Smart Grid*, 10(1):5–15, 2017.

Ramtin Madani, Morteza Ashraphijuo, and Javad Lavaei. Promises of conic relaxation for contingency-constrained optimal power flow problem. *IEEE Transactions on Power Systems*, 31(2):1297–1307, 2015.

Ramtin Madani, Mohsen Kheirandishfard, Javad Lavaei, and Alper Atamtürk. Penalized semidefinite programming for quadratically-constrained quadratic optimization. *Journal of Global Optimization*, 78:423–451, 2020.

Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7195–7201. IEEE, 2016.

Gábor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358, 1998.

Somayeh Sojoudi and Javad Lavaei. Exactness of semidefinite relaxations for nonlinear optimization problems with underlying graph structure. *SIAM Journal on Optimization*, 24(4):1746–1778, 2014.

Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:580607, 2021.

Alex L Wang. *On Quadratically Constrained Quadratic Programs and their Semidefinite Program Relaxations*. PhD thesis, University of Waterloo, 2022.

Alex L Wang and Fatma Kılınç-Karzan. Accelerated first-order methods for a class of semidefinite programs. *Mathematical Programming*, pages 1–54, 2024.

Bing-Chuan Wang, Han-Xiong Li, Yun Feng, and Wen-Jing Shen. An adaptive fuzzy penalty method for constrained evolutionary optimization. *Information Sciences*, 571:358–374, 2021.

Bing-Chuan Wang, Jing-Jing Guo, Pei-Qiu Huang, and Xian-Bing Meng. A two-stage adaptive penalty method based on co-evolution for constrained evolutionary optimization. *Complex & Intelligent Systems*, 9(4):4615–4627, 2023.

Jie Wang and Liangbing Hu. Solving low-rank semidefinite programs via manifold optimization. *arXiv preprint arXiv:2303.01722*, 2023.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.

Kailai Xu, Daniel Z Huang, and Eric Darve. Learning constitutive relations using symmetric positive definite neural networks. *Journal of Computational Physics*, 428:110072, 2021.

Siyuan Xu and Minghui Zhu. Efficient gradient approximation method for constrained bilevel optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12509–12517, 2023.

Zhipan Xu and Lijun Zhang. Online non-convex learning in dynamic environments. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Victor M Zavala and Mihai Anitescu. Real-time nonlinear optimization as a generalized equation. *SIAM Journal on Control and Optimization*, 48(8):5444–5467, 2010.

Fariba Zohrizadeh, Mohsen Kheirandishfard, Edward Quarm Jnr, and Ramtin Madani. Penalized parabolic relaxation for optimal power flow problem. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1616–1623. IEEE, 2018.

Appendix A. Missing proofs in the main text

A.1. Proof of Lemma 1

Establishing the Descent Condition: From Algorithm 1 in Xu and Zhu (2023), the line search ensures that the step size σ_k satisfies the sufficient decrease condition:

$$G\left(W_{k+1}\right) = G\left(W_k - \sigma_k g^k\right) < G\left(W_k\right) - \beta\sigma_k \left\|g^k\right\|^2,$$

where $\beta \in (0,1)$ line search parameter controls the sufficient decrease condition and g^k is the descent direction computed at iteration k. It is either the gradient $\nabla G\left(W_k\right)$ when $X^*(W_k)$ is differentiable in a neighborhood of W_k, or the vector with the smallest norm in the convex hull $\operatorname{conv}\mathcal{H}\left(W_k, \varepsilon_k\right)$ which is closed, bounded and convex. Then, when $X^*(W_k)$ is not differentiable.

$$g^k = \underset{g \in \operatorname{conv}\mathcal{H}(W_k, \varepsilon_k)}{\arg\min} \|g\|. \tag{10}$$

Summing the Descent over Iterations: Summing the sufficient decrease condition over iterations $k = 1$ to K:

$$\beta \sum_{k=1}^{K} \sigma_k \left\|g^k\right\|^2 < G\left(W_1\right) - G\left(W_{K+1}\right), \tag{11}$$

note that $G\left(W_{K+1}\right) \geq G_{\text{inf}}$, where G_{inf} is the lower bound of G

Establishing a Lower Bound on Step Sizes σ_k: To proceed, we need to ensure that the step sizes σ_k are bounded below by a positive constant $\sigma_{\text{min}} > 0$ for all k. From the proof of Theorem 3 in Xu and Zhu (2023), particularly Part (ii.b), it's shown by contradiction that σ_k does not converge to zero. The argument is as follows: Suppose that $\sigma_k \to 0$. This would imply that W_k converges to a point $\bar{W}$, and $\sigma_k \left\|g^k\right\| \to 0$, since $W_{k+1} = W_k - \sigma_k g^k$. However, due to the properties of the line search and the sufficient decrease condition, there exists a positive lower bound σ_{min} such that $\sigma_k \geq \sigma_{\text{min}}$ for all k. This is because, when σ_k is sufficiently small, the quadratic approximation of G ensures that the sufficient decrease condition cannot be satisfied unless σ_k is bounded away from zero. Therefore, we have:

$$\sigma_k \geq \sigma_{\text{min}} > 0 \quad \text{for all } k$$

Deriving the Average Gradient Norm: Using the lower bound σ_{min} :

$$\beta\sigma_{\text{min}} \sum_{k=1}^{K} \left\|g^k\right\|^2 \leq \beta \sum_{k=1}^{K} \sigma_k \left\|g^k\right\|^2 < G\left(W_1\right) - G\left(W_{K+1}\right).$$

Next, we can write it as:

$$\beta\sigma_{\text{min}} \sum_{k=1}^{K} \left\|g^k\right\|^2 < G\left(W_1\right) - G\left(W_K\right).$$

Divide both sides by K and taking the square root of both sides and applying Jensen's inequality (since the square root is concave):

$$\frac{1}{K}\sum_{k=1}^{K} \left\|g^k\right\| \leq \sqrt{\frac{1}{K}\sum_{k=1}^{K} \|g^k\|^2} \leq \sqrt{\frac{G\left(W_1\right) - G\left(W_K\right)}{\beta\sigma_{\text{min}} K}}$$

Thus, we have derived:

$$\frac{1}{K}\sum_{k=1}^{K}\left\|g^{k}\right\| \leq \sqrt{\frac{G\left(W_{1}\right)-G\left(W_{K}\right)}{\beta \sigma_{\min} K}}.$$

Solving for K : Suppose we desire the average gradient norm to be less than a threshold δ :

$$K \geq \frac{G\left(W_{1}\right)-G\left(W_{K}\right)}{\beta \sigma_{\min} \delta^{2}}. \tag{12}$$

From Proposition 3 in Xu and Zhu (2023), $G(W)$ is Lipschitz continuous on $\mathcal{B}(W, \epsilon)$. Specifically, there exists a Lipschitz constant $L_{G}=l_{G}(W, \epsilon)$ such that:

$$\left|G(W)-G\left(W^{\prime}\right)\right| \leq L_{G}\left\|W-W^{\prime}\right\|, \quad \forall W^{\prime} \in \mathcal{B}(W, \epsilon).$$

We consider the sequence $\{W_{k}\}$ generated by the algorithm and utilize the local Lipschitz continuity in each step. For each iteration k, there exists a neighborhood $\mathcal{B}\left(W_{k}, \epsilon_{k}\right)$ such that $W_{k+1} \in \mathcal{B}\left(W_{k}, \epsilon_{k}\right)$, and $G(W)$ is Lipschitz continuous on $\mathcal{B}\left(W_{k}, \epsilon_{k}\right)$ with Lipschitz constant $L_{G}, k = l_{G,k}\left(W_{k}, \epsilon_{k}\right)$. Thus, for each k:

$$G\left(W_{k}\right)-G\left(W_{k+1}\right) \leq L_{G,k}\left\|W_{k+1}-W_{k}\right\|,$$

Let $L_{\max}$ be the maximum Lipschitz constant along the path:

$$L_{\max}=\max_{1 \leq k \leq K} L_{G,k}. \tag{13}$$

Summing the inequalities over $k=1$ to $K-1$:

$$\begin{aligned} G\left(W_{1}\right)-G\left(W_{K}\right) &= \sum_{k=1}^{K-1}\left[G\left(W_{k}\right)-G\left(W_{k+1}\right)\right] \\ &\leq \sum_{k=1}^{K-1} L_{k}\left\|W_{k+1}-W_{k}\right\| \\ &\leq L_{\max} \sum_{k=1}^{K-1}\left\|W_{k+1}-W_{k}\right\| \\ &= L_{\max}\left\|W_{K}-W_{1}\right\|. \end{aligned} \tag{14}$$

Then, we can choose K to satisfy the following inequality

$$\frac{G\left(W_{1}\right)-G\left(W_{K}\right)}{\beta \sigma_{\min} \delta^{2}} \leq K \leq \frac{L_{\max}\left\|W_{K}-W_{1}\right\|}{\beta \sigma_{\min} \delta^{2}}. \tag{15}$$

The lower bound ensures you perform enough iterations to achieve the desired average gradient norm δ. The upper bound prevents you from overestimating the required iterations.

A.2. Proof of Lemma 2

The dynamic regret is defined as:

$$R_T^{\mathrm{d}} = \sum_{t=1}^{T} \ell_t\left(\theta_t\right) - \sum_{t=1}^{T} \ell_t\left(\theta_t^*\right) \tag{16}$$

The meta loss is deifned as:

$$\ell_t(\theta_t) = \left\|\mathcal{M}_{\theta_t}\left(\phi_t\right) - W_{t,K}\right\|^2 \tag{17}$$

We define empirical task-relatedness as $\mathcal{S}_{W^*}^2 = \frac{1}{T}\sum_{t=1}^{T}\left\|\mathcal{M}_{\theta_t^*}\left(\phi_t\right) - W_t^*\right\|^2$ with respect to a sequence of changing comparator optimal solution $\{W_t^*\}_{t=1}^{T}$. This measure quantifies the average discrepancy between the meta-model's predictions $\mathcal{M}_{\theta_t^*}\left(\phi_t\right)$ after K iterations of optimization and the true optimal solutions W_t^* across all tasks. The average squared initialization error is:

$$\frac{1}{T}\sum_{t=1}^{T}\left\|\mathcal{M}_{\theta_t}\left(\phi_t\right) - W_{t,K}\right\|^2 = \frac{R_T^{\mathrm{d}}}{T} + \mathcal{S}_{W^*}^2 \tag{18}$$

A.2.1. CONVEX LOSSES FUNCTION:

For online gradient descent algorithm, the dynamic regret for convex losses in θ can be bounded from (Besbes et al., 2015, Theorem. 3) as:

$$R_T^{\mathrm{d}} \leq \mathcal{C}_1 V_T^{1/3} T^{2/3}, \tag{19}$$

where $V_N = \sum_{t=2}^{T}\left\|\theta_t^* - \theta_{t-1}^*\right\|$ is the path length, $\mathcal{C}_1$ is a constant that depends on the gradient bound $D = \max_{\theta_1,\theta_2 \subset \Theta}\|\theta_1 - \theta_2\|_\infty$, and $\|\nabla \ell_t(\theta)\| \leq \ell, \forall \theta \in \Theta, \forall t$ is a bound on the gradient norms. So the average squared initialization error is bounded as:

$$\frac{1}{T}\sum_{t=1}^{T}\left\|\mathcal{M}_{\theta_t}\left(\phi_t\right) - W_{t,K}\right\|^2 = \frac{R_T^{\mathrm{d}}}{T} \leq \frac{\mathcal{C}_1 V_T^{1/3} T^{2/3}}{T} + \mathcal{S}_{W^*}^2, \tag{20}$$

From your earlier convergence analysis, the number of iterations K_t for task t is bounded by:

$$K_t \leq \frac{L_{\max}\left\|\mathcal{M}_{\theta_t}\left(\phi_t\right) - W_{t,K}\right\|}{\beta \sigma_{\min} \delta^2}$$

The expected total number of iterations over T tasks:

$$\frac{1}{T}\sum_{t=1}^{T} K_t \leq \frac{L_{\max}\sum_{t=1}^{T}\left(\left\|\mathcal{M}_{\theta_t}\left(\phi_t\right) - W_{t,K}\right\|\right)}{\beta \sigma_{\min} \delta^2 T} \tag{21}$$

Using (20), Cauchy-Schwarz, and for non-negative a and b $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ inequalities

$$\frac{1}{T}\sum_{t=1}^{T} K_t \leq \frac{L_{\max}}{\beta \sigma_{\min} \delta^2}\left(\sqrt{\mathcal{C}_1} V_T^{1/6} T^{-1/6} + \mathcal{S}_{W^*}\right) \tag{22}$$

A.2.2. NON-CONVEX LOSSES FUNCTION:

Using follow-the-perturbed learning-A algorithm proposed in Xu and Zhang (2024), the dynamic regret for non-convex losses in θ can be bounded from (Xu and Zhang, 2024, Theorem. 4) as:

$$R_T^{\mathrm{d}} \leq \mathcal{O}\left((1+\alpha_k\sqrt{T}+\gamma_k T)T^{\frac{2}{3}}\left(V_T+1\right)^{\frac{1}{3}}\right), \tag{23}$$

when $\alpha_k = O(1/\sqrt{T})$ and $\gamma_k = O(1/T)$ in addition to some constant representing D and G, R_T^{d} can be bounded as:

$$R_T^{\mathrm{d}} \leq \mathcal{C}_2 V_T^{1/3} T^{2/3}, \tag{24}$$

then following the same procedure similar to Section. (A.2.1).

Appendix B. Experiment details

B.1. Problem formulation:

The OPF problem is formulated as follows Bose et al. (2015):

$$\begin{aligned}
&\text{minimize} V^H C_0 V,\\
&\text{subject to } \underline{P}_k \leq V^H \Phi_k V \leq \bar{P}_k, \quad k = 1,\dots,n,\\
&\qquad \underline{Q}_k \leq V^H \Psi_k V \leq \bar{Q}_k, \quad k = 1,\dots,n,\\
&\qquad \underline{W}_k \leq V^H J_k V \leq \bar{W}_k, \quad k = 1,\dots,n,\\
&\qquad V^H M^{ij} V \leq \bar{F}_{ij}, \quad i \sim j,\\
&\qquad V^H T^{ij} V \leq \bar{L}_{ij}, \quad i \sim j,\\
&\qquad X = VV^H \succeq 0,
\end{aligned} \tag{25}$$

where $V \in \mathbb{C}^n$ represents the voltage vector of the buses, where each entry corresponds to the complex voltage at a bus in the network. The matrix C_0 defines the objective function, which can represent various goals such as minimizing power losses or production costs. The Hermitian matrices Φ_k and Ψ_k capture the real and reactive power injection constraints at node k, respectively, while the diagonal matrix J_k ensures the voltage magnitude constraints at the same node. Additionally, the matrices M^{ij} and T^{ij} encode the real power flow and thermal losses, respectively, for the transmission line connecting nodes i and j. The notation $i \sim j$ is used to indicate that nodes i and j are connected by a transmission line. Each of these matrices captures network properties based on the system's admittance matrix Y, defined as:

$$Y_{ij} = \begin{cases} y_{ii} + \sum_{j\sim i} y_{ij}, & \text{if } i = j\\ -y_{ij}, & \text{if } i \neq j \text{ and } i \sim j\\ 0, & \text{otherwise,}\end{cases}$$

where $y_{ij} = g_{ij} - \mathbf{i}b_{ij}$ represents the admittance between connected nodes i and j, with $g_{ij} \geq 0$ and $b_{ij} \geq 0$. To facilitate convex relaxation, the problem is reformulated using the trace operator as follows:

$$\begin{array}{ll}\text{minimize} & \operatorname{Tr}\left(C_0 X\right), \\ \text{subject to} & \underline{P}_k \leq \operatorname{Tr}\left(\Phi_k X\right) \leq \bar{P}_k, \quad k=1,\ldots,n, \\ & \underline{Q}_k \leq \operatorname{Tr}\left(\Psi_k X\right) \leq \bar{Q}_k, \quad k=1,\ldots,n \\ & \underline{W}_k \leq \operatorname{Tr}\left(J_k X\right) \leq \bar{W}_k, \quad k=1,\ldots,n, \\ & \operatorname{Tr}\left(M^{ij} X\right) \leq \bar{F}_{ij}, \quad i \sim j, \\ & \operatorname{Tr}\left(T^{ij} X\right) \leq \bar{L}_{ij}, \quad i \sim j, \\ & X \succeq 0, \quad \operatorname{rank}(X)=1,\end{array}$$

where $X = VV^H$ is a positive semidefinite matrix capturing the quadratic dependence of the objective and constraints on the voltage vector V. This reformulation transforms the original quadratic constraints into linear constraints with respect to X. To compute the (i, j) th entries of these matrices, we use the following relations for $1 \leq k \leq n$ and (p, q), (i, j) in the network graph $\mathcal{G}$:

$$[\Phi_k]_{ij} = \begin{cases} \frac{1}{2}Y_{ij} = \frac{1}{2}\left(-g_{ij} + \mathbf{i}b_{ij}\right), & \text{if } k = i, \\ \frac{1}{2}Y_{ij}^{\mathcal{H}} = \frac{1}{2}\left(-g_{ij} - \mathbf{i}b_{ij}\right), & \text{if } k = j, \\ 0, & \text{otherwise;} \end{cases}$$

$$[\Psi_k]_{ij} = \begin{cases} \frac{-1}{2\mathbf{i}}Y_{ij} = \frac{1}{2}\left(-b_{ij} - \mathbf{i}g_{ij}\right), & \text{if } k = i, \\ \frac{1}{2\mathbf{i}}Y_{ij}^{\mathcal{H}} = \frac{1}{2}\left(-b_{ij} + \mathbf{i}g_{ij}\right), & \text{if } k = j, \\ 0, & \text{otherwise;} \end{cases}$$

$$[M^{pq}]_{ij} = \begin{cases} g_{pq}, & \text{if } i = j = p, \\ \frac{1}{2}\left(-g_{pq} + \mathbf{i}b_{pq}\right), & \text{if } (i,j) = (p,q), \\ \frac{1}{2}\left(-g_{pq} - \mathbf{i}b_{pq}\right), & \text{if } (i,j) = (q,p), \\ 0, & \text{otherwise;} \end{cases}$$

$$[T^{pq}]_{ij} = \begin{cases} g_{pq}, & \text{if } i = j = p, \\ -g_{pq}, & \text{if } i = j = q, \\ 0, & \text{if } (i,j) = (p,q) = (q,p), \\ 0, & \text{otherwise.} \end{cases}$$

B.2. Meta model

The meta-learning model $\mathcal{M}_{\theta_t}$ in Optimal Power Flow case study (5.2), which predicts effective initializations based on problem-specific features ϕ_t. This model, implemented as a Hermitian neural network. The real part of the Hermitian matrix is generated by processing the input features through a feed-forward neural network with ReLU activation functions. This branch includes a final linear layer followed by a custom `spdlayers.Eigen` layer proposed in Xu et al. (2021) to ensure that the output matrix is symmetric. Similarly, the imaginary part is generated through a parallel feed-forward neural network with ReLU activations. After the final linear layer, a `spdlayers.Eigen` layer produces a symmetric matrix, which is then adjusted in the forward pass to enforce skew-symmetry. This is achieved by extracting the lower triangular part of the output, negating it, and combining it with the upper triangular part to construct a skew-symmetric imaginary component.