Compact Optimality Verification for Optimization Proxies

Wenbo Chen [1 2] **Haoruo Zhao** [1 2] **Mathieu Tanneau** [1 2] **Pascal Van Hentenryck** [1 2]

Abstract

Recent years have witnessed increasing interest in optimization proxies, i.e., machine learning models that approximate the input-output mapping of parametric optimization problems and return near-optimal feasible solutions. Following recent work by (Nellikkath & Chatzivasileiadis, 2021), this paper reconsiders the optimality verification problem for optimization proxies, i.e., the determination of the worst-case optimality gap over the instance distribution. The paper proposes a compact formulation for optimality verification and a gradient-based primal heuristic that brings substantial computational benefits to the original formulation. The compact formulation is also more general and applies to non-convex optimization problems. The benefits of the compact formulation are demonstrated on large-scale DC Optimal Power Flow and knapsack problems.

1. Introduction

In recent years, there has been a surge of interest in optimization proxies, i.e., differentiable programs that approximate the input/output mappings of parametric optimization problems. Parametric optimization problems arise in many application areas, including power systems operations, supply chain management, and manufacturing. To be applicable in practice, these optimization proxies need to satisfy two key properties: they must return *feasible* solutions and they must return *high-quality* solutions. Feasibility has received significant attention in recent years. Constraint violations can be mitigated by incorporating them in the loss function (Fioretto et al., 2020; Tran et al., 2021; Velloso & Van Hentenryck, 2021; Pan et al., 2020). In some cases, the feasibility could be guaranteed by designing mask operations in the autoregression decoding (Bello et al., 2016; Khalil et al., 2017; Song et al., 2022) or designing projection procedures (Joshi et al., 2019; Amos & Kolter, 2017; Agrawal et al., 2019; Tordesillas et al., 2023; Park et al., 2023; Li et al., 2023; Chen et al., 2023a). For better efficiency, the projection procedure can be approximated with unrolled first-order methods (Chen et al., 2020; Donti et al., 2021; Scieur et al., 2022; Monga et al., 2021; Sun et al., 2022).

However, quality guarantees for optimization proxies have received much less attention and have remained largely empirical. To the authors' knowledge, (Nellikkath & Chatzivasileiadis, 2021) is the only work that provably provides worst-case guarantees for optimization proxies. They formalize the optimality verification problem as a bilevel optimization and reformulate it into a single level using traditional techniques from optimization theory. However, the resulting single-level formulation faces significant computational challenges as it introduces a large number of auxiliary decision variables and constraints. Moreover, it only applies to convex parametric optimization problems.

This paper reconsiders the optimality verification problem and proposes a compact formulation that applies to non-convex problems and is more advantageous from a computational standpoint. Moreover, the paper proposes an effective primal heuristic, based on a (parallelized) Projected Gradient Attack (PGA). When the problem is convex, PGA makes use of a conservative approximation of the value function of the optimization that leverages subgradients. To showcase the benefits of these modeling and algorithmic contributions, the paper considers two applications: the DC Optimal Power Flow (DC-OPF) problems and the knapsack problems. The DC-OPF illustrates how the compact optimality verification formulation and PGA can be applied to realistic industrial-sized test cases, while the knapsack problem shows their applicability to a non-convex problem. The paper also proposes novel Mixed-Integer Programming (MIP) encoding of the feasibility layers of the DC-OPF and knapsack optimization proxies. Experimental results are provided to highlight the benefits of the compact optimality verification formulation.

The contributions of this paper are summarized as follows.

- The paper proposes a *compact formulation for the opti-*

[1]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, USA [2]NSF Artificial Intelligence Research Institute for Advances in Optimization (AI4OPT), USA. Correspondence to: Wenbo Chen <wenbo.chen@gatech.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

mality verification of optimization proxies. The formulation provides computational benefits compared to the existing bilevel formulation and can be used to verify non-convex optimization problems.

- The paper proposes a (parallelized) Projected Gradient Attack (PGA) that is effective in finding high-quality feasible solutions i.e., adversary points. The PGA features a novel conservative approximation of convex value functions.
- The paper demonstrates how to apply the compact optimality verification to two problems: large-scale DC Optimal Power Flows (DC-OPF) and knapsack problems. The paper contributes new feasibility restoration layers for the proxies and new encoding of these layers as MIP models for verification purposes.
- Extensive experiments show the compact formulation, together with the primal heuristic, can effectively verify optimization proxies for large-scale DC-OPF problems and knapsack problems. The compact formulation also brings substantial computational benefits compared to the bilevel formulation.

The rest of the paper is organized as follows. Section 2 surveys the relevant literature. Section 3 introduces background knowledge about parametric optimization, optimization proxies, and optimality verification. Section 4 presents the compact formulation for the optimality verification. Section 5 discusses the projected gradient attack with value function approximation. Section 6 presents the optimality verification on DC-OPF proxies and reports the numerical results on realistic industrial-size instances. Section 7 presents the optimality verification for non-convex parametric optimization problems using the knapsack problem as a case study. Section 8 concludes the paper and discusses future research directions. Section 8 includes the potential broader impact of the work.

2. Related Work

Neural networks have been increasingly used in safety-critical applications such as autonomous driving (Bojarski et al., 2016), aviation (Julian et al., 2016) and power systems (Chen et al., 2022b). However, it has become apparent that they may be highly sensitive to adversarial examples (Szegedy et al., 2013) i.e., a small input perturbation could cause a significant and undesired change in the output. To characterize the effects of such perturbations, recent verification algorithms typically employ one of two strategies: either they perform an exhaustive search of the input domain to identify a worst-case scenario (Venzke et al., 2020; Nellikkath & Chatzivasileiadis, 2021), or they adopt a less computationally intensive method to approximate an upper bound for the worst-case violation (Chen et al., 2022a; Xu et al., 2021; Raghunathan et al., 2018).

While optimality verification falls in the first category, it departs from the large body of the literature in two ways. First, evaluating the objective of the optimality verification problem involves the solving of an optimization problem, which brings unique modeling and computational challenges. Second, motivated by practical application in power systems, this paper considers larger and more complex input perturbations than the small ℓ_p-norm perturbation often considered in existing work, as outlined in (Wang et al., 2021).

It is also important to mention the work on dual optimization proxies (Qiu et al., 2023). Dual optimization proxies provide dual feasible solutions at inference time for a particular instance. In contrast, optimality verification provides a worst-case guarantee for a distribution of instances. They complement each other naturally. (Qiu et al., 2023) show how to derive dual optimization proxies for the second-order cone relaxation of AC-OPF.

3. Preliminaries

3.1. Parametric Optimization

Consider a parametric optimization problem of the form

$$P(\boldsymbol{x}): \quad \min_{\boldsymbol{y}} \quad c(\boldsymbol{x}, \boldsymbol{y}) \tag{1a}$$

$$\text{s.t.} \quad g(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \tag{1b}$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is the input *parameter*, $\boldsymbol{y} \in \mathbb{R}^n$ is the decision variable, $c : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$ is the cost function and $g : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^m$ represents the constraints. The *feasible set* is denoted by $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n \mid g(\boldsymbol{x}, \boldsymbol{y}) \leq 0\}$. The *optimal value* of $P(\boldsymbol{x})$ is denoted by $\Phi(\boldsymbol{x})$ and Φ is referred to as the *value function*. The set of optimal solutions is denoted by $\mathcal{Y}^*(\boldsymbol{x}) \subseteq \mathcal{Y}(\boldsymbol{x})$. The parametric optimization problem can be viewed as a mapping from the input parameter $\boldsymbol{x} \in \mathcal{X}$ to an optimal decision $\boldsymbol{y}^* \in \mathcal{Y}^*(\boldsymbol{x})$.

For instance, in the Optimal Power Flow (OPF) problem, the parameters are the electricity demand and generation costs. The optimization consists in finding the most cost-effective power generation that satisfies the physical and engineering constraints. In the Traveling Salesman Problem (TSP), the parameters are the locations and travel costs and the optimization consists in finding the shortest possible route that visits each location exactly once and returns to the starting point.

In a minimization problem, a feasible solution provides a primal bound, which is an upper bound on the optimal value. A dual bound is obtained from a dual-feasible solution and/or via branch-and-bound, and establishes a lower bound on the optimal value. Dual bounds are central to proving global optimality. Note that, when maximizing, a

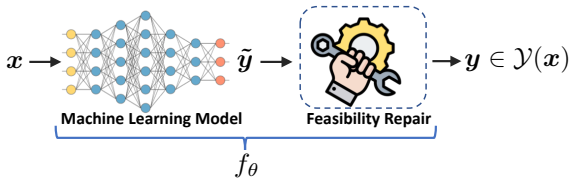


Figure 1. Optimization Proxies

primal (resp dual) bound is a lower (resp. upper) bound on the optimal value.

3.2. Optimization Proxies

Optimization proxies (see Figure 1) are differentiable programs that approximate the mapping from input parameters $\boldsymbol{x}$ to optimal decisions $\boldsymbol{y}^*$ of the parametric optimization problem. Because outputs of machine learning typically cannot satisfy complex constraints, optimization proxies consist of two parts, a machine learning model predicting the optimal decision and a feasibility repair step that projects the prediction into the feasible space. Trained optimization proxies are denoted by f_θ, with θ denoting the weights.

There has been significant progress in ensuring that optimization proxies produce feasible solutions e.g., (Li et al., 2023; Donti et al., 2021; Chen et al., 2023a). Therefore, this paper assumes that optimization proxies provide feasible solutions, i.e.,

$$\forall \boldsymbol{x} \in \mathcal{X},\ f_\theta(\boldsymbol{x}) \in \mathcal{Y}(\boldsymbol{x}). \tag{2}$$

Depending on the parametrization of the machine learning models and the modeling of the feasibility repair steps, optimization proxies could be trained using supervised learning (Joshi et al., 2019; Kotary et al., 2022; 2021; Chen et al., 2022b), reinforcement learning (Bello et al., 2016; Khalil et al., 2017; Song et al., 2022) and unsupervised learning (Karalias & Loukas, 2020; Wang et al., 2022; Donti et al., 2021; Park & Van Hentenryck, 2023; Chen et al., 2023a).

3.3. Optimality Verification

Let $\mathcal{X} \subseteq \mathbb{R}^p$ denote the set of possible inputs and f_θ be an optimization proxy. The *Optimality Verification Problem* can be formalized as

$$(P_o): \quad \max_{\boldsymbol{x} \in \mathcal{X}} \quad c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) - \Phi(\boldsymbol{x}) \tag{3}$$

Since the proxy is feasible, i.e., $f_\theta(\boldsymbol{x}) \in \mathcal{Y}(\boldsymbol{x})$, it follows that $c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) \geq \Phi(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{X}$. In general, there is no analytical formula for Φ, which is typically not differentiable or even continuous. Hence P_o cannot be solved directly.

(Nellikkath & Chatzivasileiadis, 2021) introduced a bilevel formulation for the optimality verification problem

$$\max_{\boldsymbol{x} \in \mathcal{X}} \quad c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) - c(\boldsymbol{x}, \boldsymbol{y}^*) \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{y}^* \in \arg\min_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \quad c(\boldsymbol{x}, \boldsymbol{y}). \tag{4b}$$

In (4), the leader (upper level) chooses input $\boldsymbol{x} \in \mathcal{X}$ and computes the neural network output $f_\theta(\boldsymbol{x})$ via a MIP encoding. The follower (lower level) then computes an optimal solution $\boldsymbol{y}^*$ of the optimization problem (4b). When the lower level is linear or quadratic, the bilevel formulation admits a single-level reformulation using the KKT conditions and mixed-complementarity constraints. Assuming that the proxy is a ReLU-based DNN, the overall verification problem can then be cast as an Mixed Integer Linear Programming (MILP) and solved with off-the-shelf optimization solvers like Gurobi. More generally, a single-level reformulation requires strong duality assumptions and introduces additional variables and constraints, especially non-convex complementarity constraints. It creates some computational challenges as documented in the experiments. Moreover, the reformulation is not available when problem P is non-convex, since KKT conditions are not sufficient for optimality, and may not apply if, e.g., the lower level is discrete. In general, bilevel optimization with non-convex lower-level problems is Σ_2^P hard (Caprara et al., 2013). No existing solver can solve such problems efficiently.

4. Compact Optimality Verification

The core contribution of this paper is a compact formulation for the optimality verification problem, that addresses the shortcomings of the bilevel approach. Namely, the paper proposes to formulate the optimality verification as

$$(P_c) \quad \max_{\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y}} \quad c(\boldsymbol{x}, \hat{\boldsymbol{y}}) - c(\boldsymbol{x}, \boldsymbol{y}) \tag{5a}$$

$$s.t. \quad \hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{5b}$$

$$\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}). \tag{5c}$$

The main difference between (5) and (4) is that the inner problem (4b) is replaced with the simpler constraint (5c). In fact, the proposed formulation (5) is the so-called high-point relaxation (Moore & Bard, 1990) of the bilevel formulation (4). Theorem 4.1 shows that (5) has the same optimum as (4), i.e., the high-point relaxation is exact. To the authors' knowledge, this result is novel.

Theorem 4.1. *Problems* (4) *and* (5) *have same optimum.*

Proof. Recall that the compact formulation (5) is a relaxation of (4). Therefore, it suffices to show that an optimal solution to (5), denoted by $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$, is feasible for (4).

Algorithm 1 Projected Gradient Attack with Value Function Approximation (PGA-VFA)

Input: Input space $\mathcal{X}$, initial point $\boldsymbol{x}_0$, value function approximation $\widehat{\Phi}(\cdot)$, number of iterations T
for $t = 0$ **to** T **do**
$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{x}_t + \lambda \cdot \nabla_{\boldsymbol{x}} \left[c(\boldsymbol{x}_t, f_{\boldsymbol{\theta}}(\boldsymbol{x}_t)) - \widehat{\Phi}(\boldsymbol{x}_t) \right]$
$\boldsymbol{x}_{t+1} = \text{Proj}_{\mathcal{X}}(\hat{\boldsymbol{x}}_t)$
end for

By definition, $\tilde{\boldsymbol{y}} \in \mathcal{Y}(\tilde{\boldsymbol{x}})$, i.e., $\tilde{\boldsymbol{y}}$ is feasible for the lower-level problem (4b) with upper-level decision $\tilde{\boldsymbol{x}}$. Next, assume $\tilde{\boldsymbol{y}}$ is not optimal for (4b), i.e., there exists $\hat{\boldsymbol{y}} \in \mathcal{Y}(\tilde{\boldsymbol{x}})$ such that $c(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{y}}) < c(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$. By construction, $(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is feasible for (5) with objective value strictly better than $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$, which contradicts the optimality of $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$. □

The proposed compact formulation (5) has several advantages. First, it naturally supports non-convex constraints and objectives, In contrast, the standard approach of reformulating bilevel problems into a single-level problem with complementarity constraints, is not possible when the lower-level problem (4b) is non-convex. *This is the first tractable exact formulation for verifying the optimality of non-convex optimization proxies.* Second, even when P is convex and a single-level reformulation is possible, the compact formulation avoids the additional variables and constraints that come with a single-level reformulation. In particular, it eliminates the need for complementarity constraints, which are notoriously difficult to solve. This last point is demonstrated in the experiments of Section 6.

5. Projected Gradient Attack

The projected gradient attack is highly effective in finding high-quality feasible solutions, i.e., adversarial examples, for verification problems. It can be formalized as

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{x}_t + \lambda \nabla_{\boldsymbol{x}} \left[c(\boldsymbol{x}_t, f_{\boldsymbol{\theta}}(\boldsymbol{x}_t)) - \Phi(\boldsymbol{x}_t) \right], \tag{6a}$$

$$\boldsymbol{x}_{t+1} = \text{Proj}_{\mathcal{X}}(\hat{\boldsymbol{x}}_{t+1}). \tag{6b}$$

When $\mathcal{X}$ is an ℓ_p ball, the projection step in (6b) can be computed in close form.

A challenge in implementing the projected gradient attack is the computation of the gradient $\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x})$. One possible approach is to use the implicit function theorem on the KKT conditions (Amos & Kolter, 2017; Agrawal et al., 2019). However, this gradient computation involves the solving of many optimization instances, which may be computationally intensive for large-scale problems as those studied in this paper. *A second key contribution of the paper is the use in the projected gradient attack of a piece-wise linear conservative approximation of convex value function.* It builds on the following well-known result.

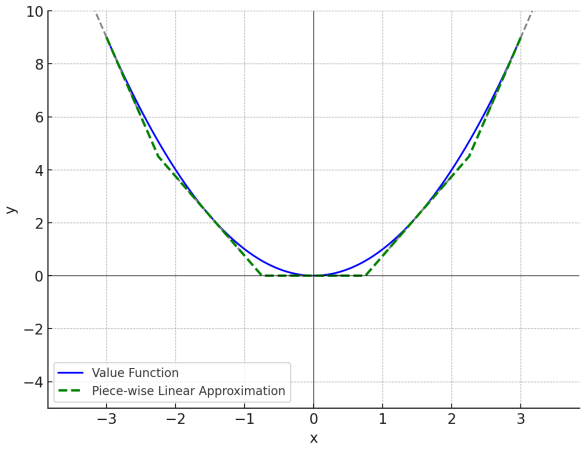


Figure 2. Value Function and its Piece-wise Linear Approximation

Theorem 5.1. *(Boyd & Vandenberghe, 2004) Consider a convex parametric optimization P where (i) the objective c and left-hand side g do not depend on $\boldsymbol{x}$ and (ii) the constraints' right-hand side is an affine function of $\boldsymbol{x}$. Then, the value function Φ of P is convex.*

Theorem 5.1 implies that a convex value function can be outer-approximated by a closed-formed piecewise linear function:

$$\widehat{\Phi}(\boldsymbol{x}) = \max_{i \in [N]} \Phi(\boldsymbol{x}_i) + \nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}_i)^{\intercal} (\boldsymbol{x} - \boldsymbol{x}_i), \tag{7}$$

where $\{\boldsymbol{x}_i\}_{i=1}^N$ denotes a set of parameters of historical parametric optimization instances. Moreover, the optimal dual $\boldsymbol{\lambda}_i$ of an instance with parameter $\boldsymbol{x}_i$ represents a subgradient of the value function with respect to $\boldsymbol{x}_i$. If the optimal dual is unique, then $\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}_i) = \boldsymbol{\lambda}_i$ (Nocedal & Wright, 1999). Therefore, the approximate value function reads:

$$\widehat{\Phi}(\boldsymbol{x}) = \max_{i \in [N]} \Phi(\boldsymbol{x}_i) + \boldsymbol{\lambda}_i^{\intercal} (\boldsymbol{x} - \boldsymbol{x}_i), \tag{8}$$

The piece-wise linear approximation on the convex value function is illustrated in Figure 2. Other models such as Input Convex Neural Networks (Amos et al., 2017) could also be used for the value function approximation.

The projected gradient attack may converge to a low-quality solution due to becoming trapped in local optima. This work proposed several techniques aimed at enhancing the search process. One acceleration technique consists in sampling a set of starting points in input domain $\{\boldsymbol{x}_0^m\}_{m=1}^M$ and run PGA-VFA($\mathcal{X}$, $\boldsymbol{x}_0^m$) for every starting point. Another acceleration technique to improve the search is to partition the input domain into subregions, and then run Algorithm (1) on different subregions $\{\mathcal{X}_p\}_{p=1}^P$ in parallel i.e., running PGA-VFA($\mathcal{X}_p$, $\boldsymbol{x}_0$) for every subregion.

6. DC Optimal Power Flow

6.1. DC-OPF Formulation

DC Optimal Power Flow (DC-OPF) is a fundamental problem for modern power system operations. It aims at determining the least-cost generator setpoints that meet grid demands while satisfying physical and operational constraints. With the penetration of renewable energy and distributed energy resources, the system operators must continuously monitor risk in real-time, i.e., they must quickly assess the system's behavior under various changes in load and renewables by solving a large volume of DC-OPF problems. However, traditional optimization solvers may not be capable of solving them quickly enough for large-scale power networks (Chen et al., 2023b). Recent advancements in learning-based methods have accelerated the process of finding feasible and empirically near-optimal solutions considerably faster than conventional approaches (Chen et al., 2023a; Zhao et al., 2022; Li et al., 2023). *This paper aims at providing formal quality guarantees for optimization proxies in this space to complement existing primal learning methods.*

Consider the DC-OPF formulation

$$\min_{\boldsymbol{p},\xi^{\text{th}}} \quad \boldsymbol{c}^\top \boldsymbol{p} + M^{\text{th}} \boldsymbol{e}^\top \xi^{\text{th}} \tag{9a}$$

$$\text{s.t.} \quad \boldsymbol{e}^\top \boldsymbol{p} = \boldsymbol{e}^\top \boldsymbol{d}, \tag{9b}$$

$$+ \boldsymbol{H}\boldsymbol{p} + \xi^{\text{th}} \geq -\bar{\boldsymbol{f}} + \boldsymbol{H}\boldsymbol{d}, \tag{9c}$$

$$- \boldsymbol{H}\boldsymbol{p} + \xi^{\text{th}} \geq -\bar{\boldsymbol{f}} - \boldsymbol{H}\boldsymbol{d}, \tag{9d}$$

$$\underline{\boldsymbol{p}} \leq \boldsymbol{p} \leq \bar{\boldsymbol{p}}, \tag{9e}$$

$$\boldsymbol{p} \in \mathbb{R}^B, \xi^{\text{th}} \in \mathbb{R}^E_+ \tag{9f}$$

where B, E denote the total number of buses and transmission lines in the power grid, respectively, $\boldsymbol{d} \in \mathbb{R}^B$ denotes the electricity load, $\boldsymbol{p}$ denotes the decision variables capturing energy dispatches, ξ^{th} denotes the thermal limit violations, $\bar{\boldsymbol{f}}$ corresponds to the flow limits on the transmission lines and $\boldsymbol{H} \in \mathbb{R}^{E \times B}$ denotes the power transfer distribution factors. The price of electricity generation is represented by $\boldsymbol{c}$, and M^{th} is the price of violating thermal constraints. The vector $\boldsymbol{e}$ consists of all ones. Constraint (9b) ensures the global power balance i.e., the total load equals the total supply. Constraint (9c) and (9d) measure the thermal violations. Constraint (9e) ensures that the outputs of the generators remain within their physical limits.

6.2. Compact Optimality Verification

This work considers the optimization proxy proposed in (Chen et al., 2023a) and illustrated in Figure 3. It consists of a fully-connected neural network with ReLU activation to predict the optimal dispatches and feasibility layers to ensure that the outputs satisfy the hard constraints. The

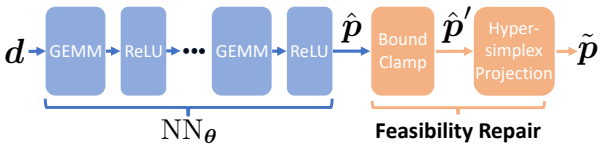


Figure 3. Optimization Proxies for DCOPF

feasibility layer consists of two parts, a bound-clamping layer, which employs a hard-sigmoid function to ensure that dispatch decisions remain within the generators' physical upper and lower generation limits (9e) and a hypersimplex layer, which uses a differentiable binary search to guarantee that total power generation matches the total electricity demand, addressing constraint (9b). Constraints (9c) and (9d) are soft and are penalized in the loss function when training the ML models. *Another contribution of this paper is a compact encoding of these layers into a mixed-integer programming formulation.* The detailed modeling of the compact formulation is deferred in Appendix A.1.

6.3. Empirical Evaluation

6.3.1. Experiment Setup

The optimality verification for DCOPF proxies is evaluated over IEEE 57-/118-/300-bus and Pegase 1354-bus test cases from the PGLib library (Babaeinejadsarookolaee et al., 2019). The data generation follows (Chen et al., 2023a). Denote by $\boldsymbol{d}^{\text{ref}}$ the nodal load vector from the reference PGLib case. The instances are generated by perturbing the reference load vector. Namely, for instance i, $\boldsymbol{d}^{(i)} = (\gamma^{(i)} + \boldsymbol{\eta}^{(i)}) \times \boldsymbol{d}^{\text{ref}}$, where $\gamma^{(i)} \in \mathbb{R}$ is a global scaling factor and $\boldsymbol{\eta}^{(i)} \in \mathbb{R}^B$ denotes element-wise noises. The γ is sampled from uniform distribution $U[80\%, 120\%]$ and for each load, η is sampled from a uniform distribution $U[-5\%, 5\%]$. This distribution captures system-wide correlations (γ), while allowing for local variability ($\boldsymbol{\eta}$). The optimization proxies are trained using the self-supervised learning algorithm in (Chen et al., 2023a). It is important to note that the verification problem only depends on the weights of the trained proxy.

The parameter input domain $\mathcal{X}$ reflects the support of the distribution of instances described above. Namely,

$$\mathcal{X} = \{(\alpha + \boldsymbol{\beta}) \cdot \boldsymbol{d}^{\text{ref}} | -u \leq \alpha - 1 \leq u, -\mathbf{5}\% \leq \boldsymbol{\beta} \leq \mathbf{5}\%\},$$

where $\alpha \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^B$ capture the distribution of $\gamma, \boldsymbol{\eta}$, and $u \in \{0, 1\%, 2\%, 5\%, 10\%, 20\%\}$ controls the size of the input domain. Each value of u yields a different optimality verification instance; note that larger values of u make the instances harder to verify.

The value function approximation $\hat{\Phi}$, used in PGA-VFA (Algorithm 1), is constructed using primal and dual solutions of 50,000 instances, generated using the above distribution.

Table 1. Comparison of presolved model size for Bilevel and Compact (proposed) formulations. Statistics are averages across 30 instances (6 distinct values of u and 5 unique seeds).

	#ConVars		#BinVars		#Constraints	
System	Bilevel	Compact	Bilevel	Compact	Bilevel	Compact
57	**255**	259	69	**64**	350	**342**
118	609	**522**	218	**100**	862	**619**
300	1798	**1320**	861	**325**	2982	**1821**
1354	6739	**4655**	7281	**1353**	15373	**6777**

PGA-VFA is executed in parallel, across 200 threads, using the acceleration techniques of Section 5. The initial step size is 10^{-3}, and is reduced by a factor 10 if no improvement is recorded over 10 iterations. Finally, PGA-VFA is stopped if no improved solution is found after 20 consecutive iterations, or a maximum of 500 iterations is reached.

All verification problems are solved with Gurobi 10.0 (Gurobi Optimization, LLC, 2023) using 16 threads and a 6-hour time limit. Preliminary experiments revealed that Gurobi struggles to find primal-feasible solutions. Therefore, unless specified otherwise, the $\boldsymbol{d}^{\text{ref}} \in \mathcal{X}$ is always passed as a warm-start to the solver. In addition, this work uses optimization-based bound tightening (Caprara & Locatelli, 2010) to improve the MILP relaxation, by tightening the input domain of ReLU neurons; see Appendix A.3. Finally, each verification instance is solved using 5 different seeds, and results are averaged using the shifted geometric mean

$$\mu_s(x_1, \cdots, x_n) = \sqrt[n]{\Pi_i(x_i + s)} - s.$$

The paper uses a shift s of 1% for optimality gaps and 1 for other values. Solving times are reported in wall clock. Experiments are conducted on dual Intel Xeon 6226@2.7GHz machines running Linux on the cluster.

6.3.2. Numerical Results

Effectiveness of Compact Formulation Table 1 reports the size (number of variables and constraints) of the compact formulation and bilevel formulation, after being presolved by Gurobi. Table 1 shows that the compact formulation results in substantially fewer binary decision variables than its bilevel counterpart, especially on large systems.

The size reduction of the compact formulation is reflected in the solving process. For ease of comparison, the verification instances are split into two groups. On the one hand, *closed* instances are instances that can be solved by at least one approach within the prescribed time limit. They include all 57-bus and 118-bus instances, as well as the 300-bus instances with $u = 0, 1\%, 2\%, 5\%$. On the other hand, *open* instances are those that cannot be solved by either formulation. They include the 300-bus instances with $u = 10\%, 20\%$, and all the 1354-bus instances.

Table 2. Comparison of solution times(s) for Bilevel and Compact (proposed) formulations across closed (i.e., solved) instances.

System	$\%u$	Bilevel	Compact	Speedup*
57	0	0.4	**0.3**	22.9%
	1	**0.6**	0.6	-3.4%
	2	0.6	**0.5**	10.4%
	5	1.9	**1.1**	75.8%
	10	**1.2**	1.2	-1.7%
	20	0.7	**0.5**	27.6%
118	0	1.7	**1.2**	43.3%
	1	3.9	**2.1**	89.7%
	2	4.7	**3.1**	53.6%
	5	28.4	**12.1**	135.2%
	10	18.7	**14.1**	32.4%
	20	76.8	**59.4**	29.3%
300	0	119.8	**63.4**	88.9%
	1	298.6	**166.0**	79.9%
	2	1302.7	**420.8**	209.5%
	5	17585.4	**10507.9**	67.4%

*Average speedup compared to bilevel formulation.

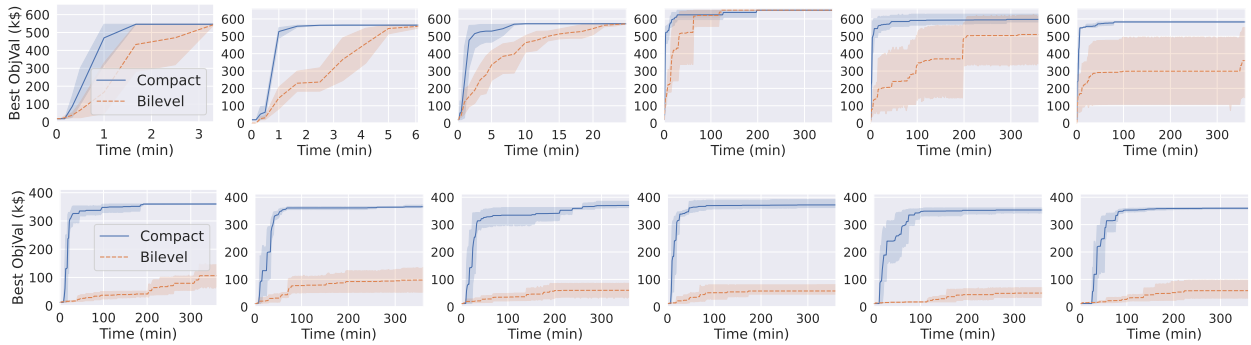
Table 3. Primal and dual objective values attained by the Bilevel and Compact formulation on open (i.e., unsolved) instances.

		Primal bound (K$)			Dual bound (M$)		
System	$\%u$	Bilevel	Compact	%Gain†	Bilevel	Compact	%Gain†
300	10	466.76	596.98	27.90	4.45	3.82	14.08
	20	410.19	582.81	42.08	10.79	9.29	13.88
1,354	0	94.22	360.03	282.13	29.64	26.00	12.25
	1	84.44	365.50	332.83	36.35	35.39	2.63
	2	50.12	369.68	637.65	47.64	45.13	5.26
	5	54.32	371.52	584.02	78.79	74.31	5.69
	10	46.63	353.20	657.42	135.32	129.99	3.94
	20	49.52	359.72	626.40	249.02	235.79	5.31

†Relative improvement compared to bilevel formulation, in %.

Table 2 reports the solving times of the compact formulation and bilevel formulation on solved instances, and indicate the relative speedup of the compact formulation, defined as $\frac{t_{\text{Bilevel}} - t_{\text{Compact}}}{t_{\text{Compact}}} \times 100\%$. The compact formulation consistently outperforms the bilevel formulation, except for two small instances which are solved by both formulations in under 2 seconds. Notably, the compact formulation achieves higher speedups on larger systems, with up to 209.5% speedup on the 300-bus system with 2% input perturbation.

Next, Table 3 reports the performance of two formulations on open instances, where none of the formulations can solve the instances to optimality within 6 hours, primarily due to the weak linear relaxation. In this case, the comparison focuses on the final primal and dual objectives. The gains in percentage under the Primal and Dual columns report the performance improvements on the primal and dual side, respectively: $\frac{\text{Primal}_{\text{Compact}} - \text{Primal}_{\text{Bilevel}}}{\text{Primal}_{\text{Bilevel}}} \times 100\%$ and $\frac{\text{Dual}_{\text{Bilevel}} - \text{Dual}_{\text{Compact}}}{\text{Dual}_{\text{Bilevel}}} \times 100\%$. As reported in Table 3, the compact formulation finds substantially better primal solutions than the bilevel formulation, especially for harder instances.



(a) $u = 0\%$ (b) $u = 1\%$ (c) $u = 2\%$ (d) $u = 5\%$ (e) $u = 10\%$ (f) $u = 20\%$

Figure 4. Evolution of primal objective value for Compact and Bilevel formulations over time on the 300 IEEE system (top row) and the 1354 Pegase system (bottom row) with different sizes of input domains. Higher values indicate better outcomes.

Table 4. Comparisons of PGA-VFA and the compact formulation on final objective and solving times for large power systems.

		PGA-VFA		Compact		
System	$\%u$	Obj. ($)	Time (s)	Obj. ($)	Time (s)	†T2PGA (s)
300	0	544923.75	4.53	546038.13	63.41	52.06
	1	562348.33	7.09	564263.39	166.01	114.79
	2	572166.01	37.30	572464.17	420.83	307.40
	5	572166.01	49.26	651163.59	10507.88	872.25
	10	572166.01	41.41	596983.07	t.o.	2646.21
	20	572166.01	17.86	582808.23	t.o.	2646.28
1,354	0	380673.60	67.26	360031.97	t.o.	t.o.
	1	399441.04	63.54	365497.51	t.o.	t.o.
	2	403725.74	41.98	369678.23	t.o.	t.o.
	5	403725.74	31.88	371524.52	t.o.	t.o.
	10	571252.51	85.88	353196.16	t.o.	t.o.
	20	571252.51	76.43	359722.30	t.o.	t.o.

†Time taken by the compact formulation to reach the same primal objective as PGA; "t.o." indicate timeouts.

Table 5. Comparisons of PGA-VFA and the compact formulation for different warm starts on final objective and solving times for large power systems

System	$\%u$	PGA-VFA	Compact+Nom	Compact+PGA
300	0	544923.75	546038.13	546038.13
	1	562348.32	564263.39	564263.44
	2	572166.01	572464.17	572464.17
	5	572166.01	651163.59	651163.38
	10	572166.01	596983.07	596983.90
	20	572166.01	582808.23	582808.47
1,354	0	380673.60	360031.97	380757.88
	1	399441.04	365497.51	399579.63
	2	403725.74	369678.23	406165.38
	5	403725.74	371524.52	406164.74
	10	571252.51	353196.16	571715.99
	20	571252.51	359722.30	578732.15

This is further supported in Figure 4, where the compact formulation consistently converges faster. On the dual side, the compact formulation is slightly better than its bilevel counterpart. Since the compact formulation systematically outperforms the bilevel formulation, the next experiments focus on the compact formulation.

Effectiveness of PGA-VFA Table 4 focuses on the performance of the proposed PGA-VFA and compact formulation in identifying feasible solutions on large power systems (300 bus and 1354 bus). The performance on small systems is deferred to Appendix A.4 since the compact formulation solves those instances within 60 seconds. PGA-VFA is particularly good at finding high-quality primal solutions within short computing time. It finds near-optimal solutions on the 300-system orders of magnitude faster than the compact formulation with a reference load vector as the warm-start. For the 300-bus system with 1% perturbation, PGA-VFA finds a feasible solution with objective 562348.33 in 7.09s. Meanwhile, Gurobi solves the compact MILP formulation in 166.01s, reporting an optimal value of 564263.39. It takes Gurobi 114.79s to find a feasible solution at least as good as the one found by PGA-VFA (in 7.09s). The compact model cannot find the same quality of primal solutions as PGA-VFA on the 1354-systems within 6 hours.

Finally, Table 5 reports the impact of warm-starting the compact formulation with PGA-VFA. The compact model with the PGA-VFA solution as a warm start always outperforms its counterparts with the reference load as the warm start. Observe also that the optimization barely improves on the PGA-VFA solution.

7. Optimality Verification for Knapsack

One major benefit of the proposed compact formulation is that it may provide quality guarantees for optimization proxies that approximate non-convex parametric optimization problems. This section illustrates this capability on the knapsack problem.

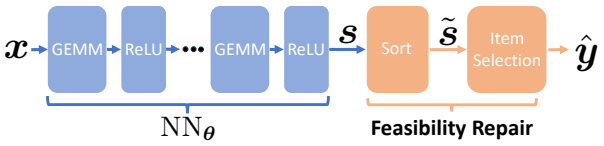


Figure 5. An Optimization Proxy for the Knapsack Problem.

7.1. Knapsack Formulation

Consider a knapsack problem with K items

$$\max_{\boldsymbol{y}} \quad \left\{\boldsymbol{v}^T\boldsymbol{y} \mid \boldsymbol{w}^T\boldsymbol{y} \leq l, \boldsymbol{y} \in \{0,1\}^K\right\}, \qquad (10)$$

where $\boldsymbol{v}$ denotes the value of the items, l is the knapsack capacity and $\boldsymbol{w}$ denotes the weight of the items. The binary decision variable $\boldsymbol{y}_k = 1$ indicates putting the item k in the knapsack and vice versa.

7.2. Optimality Verification

This paper considers the optimality verification of an optimization proxy for the knapsack problem. The proxy is summarized in Figure 5. First, a fully-connected neural network with ReLU activation predicts a score for each item, where a higher score indicating higher desirability of the item. Then, a novel feasibility repair step sorts the items by the predicted score in a descending order and adds items to the knapsack following the order until reaching the knapsack's capacity limit, thus enforcing $\boldsymbol{w}^T\boldsymbol{y} \leq l$. *Another contribution of this paper is a compact formulation of this repair layer that is presented in Appendix B.1*. Because the parametric optimization is non-convex, the bilevel formulation in (4) cannot be reformulated into a single level.

7.3. Numerical Evaluation

The data for the experiments is generated by perturbing the total capacity of the knapsack and the item values, where the scaler l of the total capacity and the item values $\boldsymbol{v}$ are sampled from the uniform distribution $U[80\%, 120\%]$. More information of the knapsack proxies is detailed in the Appendix B. The input domain is defined as:

$$\mathcal{X} = \{\alpha l, \boldsymbol{\beta v} \mid -u \leq \alpha - 1 \leq u, -\mathbf{u} \leq \boldsymbol{\beta} - 1 \leq \mathbf{u}\},$$

where u controls the size of the input domain.

Table 6 demonstrates the compact formulation can verify optimization proxies for Knapsack. *It highlights a key benefit of the compact formulation: its ability to verify non-convex optimization problems.*

Table 6. Optimality Verification for Knapsack

#items	%u	Gap (%)	Time (s)
10	1	0.0	0.4
	5	0.0	0.4
	10	0.0	0.4
	20	0.0	0.7
50	1	0.0	89.1
	5	0.0	117.3
	10	0.0	642.8
	20	142.0	t.o.
80	1	0.0	1928.3
	5	0.0	5997.8
	10	230.0	t.o.
	20	1350.0	t.o.

* Solved with 16 threads and time limits of 6 hours.

8. Conclusion

The paper presents a novel compact formulation for optimality verification of optimization proxies. It offers substantial computational benefits over the traditional bilevel formulations and can verify non-convex optimization problems. The paper also introduces a Projected Gradient Attack with a value function approximation as a primal heuristic for effectively finding high-quality primal solutions. The methodology is applied to large-scale DC-OPF and knapsack problems, incorporating new MILP encodings for the feasibility layers. Extensive experiments demonstrate the efficacy of the methodology in verifying proxies for DC-OPF and knapsack problems, highlighting its computational advantages. Future works will investigate the scalability of the methodology on large industrial instances with the coupling with spatial branch and bound and α, β-CROWN, and extend the verification on auto-regression-based optimization proxies.

Limitations The proposed compact formulation is more general than prior state-of-the-art, and has fewer variables and constraints. Nevertheless, this exact verification scheme, which eventually relies on solving MIP problems, shares common limitations with existing exact verification methods, as highlighted in (Wang et al., 2021; Zhang et al., 2022; Ferrari et al., 2022). State-of-the-art exact verification solvers primarily focus on ReLU networks, and do not always support arbitrary linear/discrete/nonlinear constraints which are needed for optimality verification. Thus, MIP solvers are the only existing tools capable of solving optimality verification problems. MIP solvers are known to struggle when solving large verification instances, especially for finding high-quality solutions (which correspond to adversarial examples). Therefore, scalability issues still exist when solving large-scale verification problems, especially with deep neural networks and large input space. Exact optimality verification on large-scale industrial instances, and exploring its effectiveness for nonlinear activations and non-perceptron architectures, e.g.,Transformers, are promising research directions.

Acknowledgements

This research was partially supported by NSF awards 2007164 and 2112533, and ARPA-E PERFORM award DE-AR0001280.

Impact Statement

Optimization proxies have been increasingly used in critical infrastructures such as power systems and supply chains, offering the potential to catalyze substantial advancements. These advancements include facilitating the transition of power systems to high-renewable grids, as well as enhancing the operational resilience and sustainability of supply chain operations. In this context, ensuring the quality and reliability of optimality proxies is essential for their practical deployment. The compact optimality verification and the gradient-based primal heuristic play a pivotal role in addressing these needs. By offering a more reliable foundation for the deployment of optimization proxies, this research has the potential to substantially impact human life and contribute to social welfare improvement.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pp. 136–145. PMLR, 2017.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *International Conference on Machine Learning*, pp. 146–155. PMLR, 2017.

Babaeinejadsarookolaee, S., Birchfield, A., Christie, R. D., Coffrin, C., DeMarco, C., Diao, R., Ferris, M., Fliscounakis, S., Greene, S., Huang, R., et al. The power grid library for benchmarking ac optimal power flow algorithms. *arXiv preprint arXiv:1908.02788*, 2019.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P., and Mudigonda, P. K. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31, 2018.

Caprara, A. and Locatelli, M. Global optimization problems and domain reduction strategies. *Mathematical Programming*, 125:123–137, 2010.

Caprara, A., Carvalho, M., Lodi, A., and Woeginger, G. J. A complexity and approximability study of the bilevel knapsack problem. In *Integer Programming and Combinatorial Optimization: 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings 16*, pp. 98–109. Springer, 2013.

Chen, S., Wong, E., Kolter, J. Z., and Fazlyab, M. Deepsplit: Scalable verification of deep neural networks via operator splitting. *IEEE Open Journal of Control Systems*, 1:126–140, 2022a. doi: 10.1109/OJCSYS.2022.3187429.

Chen, W., Park, S., Tanneau, M., and Van Hentenryck, P. Learning optimization proxies for large-scale security-constrained economic dispatch. *Electric Power Systems Research*, 213:108566, 2022b.

Chen, W., Tanneau, M., and Hentenryck, P. V. End-to-end feasible optimization proxies for large-scale economic dispatch. *IEEE Transactions on Power Systems*, pp. 1–12, 2023a. doi: 10.1109/TPWRS.2023.3317352.

Chen, W., Tanneau, M., and Van Hentenryck, P. Real-time risk analysis with optimization proxies. *arXiv preprint arXiv:2310.00709*, 2023b.

Chen, X., Li, Y., Umarov, R., Gao, X., and Song, L. Rna secondary structure prediction by learning unrolled algorithms. In *International Conference on Learning Representations*, 2020.

Donti, P. L., Rolnick, D., and Kolter, J. Z. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2021.

Ferrari, C., Mueller, M. N., Jovanović, N., and Vechev, M. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=l_amHf1oaK.

Fioretto, F., Mak, T. W., and Van Hentenryck, P. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 630–637, 2020.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Julian, K. D., Lopez, J., Brush, J. S., Owen, M. P., and Kochenderfer, M. J. Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10. IEEE, 2016.

Karalias, N. and Loukas, A. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Kotary, J., Fioretto, F., and Van Hentenryck, P. Learning hard optimization problems: A data generation perspective. *Advances in Neural Information Processing Systems*, 34:24981–24992, 2021.

Kotary, J., Fioretto, F., and Van Hentenryck, P. Fast approximations for job shop scheduling: A lagrangian dual deep learning method. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7239–7246, 2022.

Li, M., Kolouri, S., and Mohammadi, J. Learning to solve optimization problems with hard linear constraints. *IEEE Access*, 2023.

Monga, V., Li, Y., and Eldar, Y. C. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2): 18–44, 2021.

Moore, J. T. and Bard, J. F. The Mixed Integer Linear Bilevel Programming Problem. *Operations Research*, 38 (5):911–921, 1990. doi: 10.1287/opre.38.5.911.

Nellikkath, R. and Chatzivasileiadis, S. Physics-informed neural networks for minimising worst-case violations in dc optimal power flow. In *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 419–424. IEEE, 2021.

Nocedal, J. and Wright, S. J. *Numerical optimization*. Springer, 1999.

Pan, X., Zhao, T., Chen, M., and Zhang, S. Deepopf: A deep neural network approach for security-constrained dc optimal power flow. *IEEE Transactions on Power Systems*, 36(3):1725–1735, 2020.

Park, S. and Van Hentenryck, P. Self-supervised primal-dual learning for constrained optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4052–4060, 2023.

Park, S., Chen, W., Mak, T. W., and Van Hentenryck, P. Compact optimization learning for ac optimal power flow. *arXiv preprint arXiv:2301.08840*, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Qiu, G., Tanneau, M., and Van Hentenryck, P. Dual Conic Proxies for AC Optimal Power Flow. *2024 Power Systems Computation Conference, Paris*, june 2023. URL https://doi.org/10.48550/arXiv.2310.02969.

Raghunathan, A., Steinhardt, J., and Liang, P. Semidefinite relaxations for certifying robustness to adversarial examples, 2018.

Scieur, D., Gidel, G., Bertrand, Q., and Pedregosa, F. The curse of unrolling: Rate of differentiating through optimization. *Advances in Neural Information Processing Systems*, 35:17133–17145, 2022.

Song, W., Chen, X., Li, Q., and Cao, Z. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2022.

Sun, H., Shi, Y., Wang, J., Tuan, H. D., Poor, H. V., and Tao, D. Alternating differentiation for optimization layers. *arXiv preprint arXiv:2210.01802*, 2022.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

Tordesillas, J., How, J. P., and Hutter, M. Rayen: Imposition of hard convex constraints on neural networks. *arXiv preprint arXiv:2307.08336*, 2023.

Tran, C., Fioretto, F., and Van Hentenryck, P. Differentially private and fair deep learning: A lagrangian dual

approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9932–9939, 2021.

Velloso, A. and Van Hentenryck, P. Combining deep learning and optimization for preventive security-constrained dc optimal power flow. *IEEE Transactions on Power Systems*, 36(4):3618–3628, 2021.

Venzke, A., Qu, G., Low, S., and Chatzivasileiadis, S. Learning optimal power flow: Worst-case guarantees for neural networks, 2020.

Wang, H. P., Wu, N., Yang, H., Hao, C., and Li, P. Unsupervised learning for combinatorial optimization with principled objective relaxation. *Advances in Neural Information Processing Systems*, 35:31444–31458, 2022.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29909–29921. Curran Associates, Inc., 2021.

Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., and Hsieh, C.-J. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers, 2021.

Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hsieh, C.-J., and Kolter, J. Z. General cutting planes for bound-propagation-based neural network verification. *Advances in neural information processing systems*, 35:1656–1670, 2022.

Zhao, H., Hijazi, H., Jones, H., Moore, J., Tanneau, M., and Hentenryck, P. V. Bound tightening using rolling-horizon decomposition for neural network verification, 2024.

Zhao, T., Pan, X., Chen, M., and Low, S. Ensuring dnn solution feasibility for optimization problems with linear constraints. In *The Eleventh International Conference on Learning Representations*, 2022.

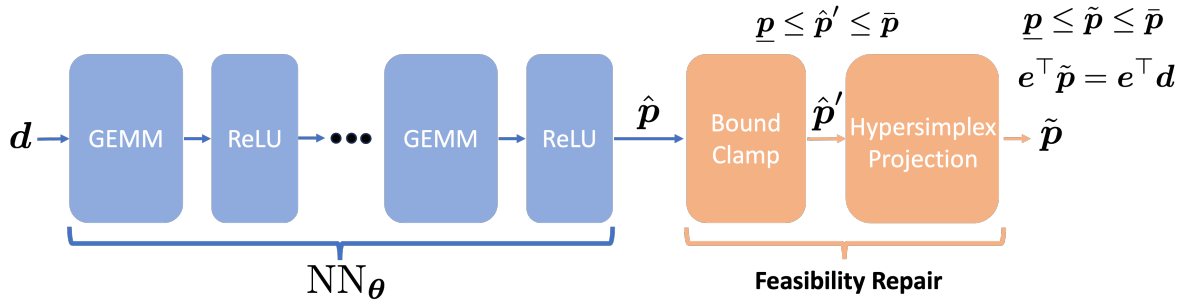


Figure 6. Optimization Proxies for DCOPF

A. More information of DCOPF Optimization Proxies

A.1. Detailed Modeling of Compact Formulation

As shown in Figure 6, the proposed DC-OPF proxy integrates a multilayer perceptron (MLP), a bound clamping, and a hypersimplex projection layer. The MLP can be translated into a Mixed Integer Linear Program (MILP) by modeling its general matrix multiplication (GEMM) blocks with linear constraints and its ReLU activations using binary variables and linear constraints, as detailed in prior works (Tjeng et al., 2019; Bunel et al., 2018).

Bound Clamp The bound clamping layer is designed to enforce minimum and maximum generation limits (9e). It can be expressed as the composition of two ReLU layers, effectively clamping the input between specified maximum and minimum power generation limits:

$$\hat{p}' = \text{clamp}(\hat{p}, \underline{p}, \bar{p}) = \min(\max(\hat{p}, \underline{p}), \bar{p}) \tag{11a}$$

$$= -\text{ReLU}(-\text{ReLU}(\hat{p} - \underline{p}) - \underline{p} + \bar{p}) + \bar{p}. \tag{11b}$$

The clamp operator clips all entries in $\hat{p}$ into the range $[\bar{p}, \underline{p}]$. It can be formulated as linear constraints and binary decision variables, leveraging the ReLU MILP representation from the aforementioned studies (Tjeng et al., 2019; Bunel et al., 2018).

Hypersimplex Projection The Hypersimplex Projection takes as input $p \in \mathcal{H} = \{p \in \mathbb{R}^B | \underline{p} \le p \le \bar{p}\}$, and outputs $\tilde{p} \in \mathcal{S}$, where $\mathcal{S}$ is the hypersimplex

$$\mathcal{S} = \{p \in \mathbb{R}^B | \underline{p} \le p \le \bar{p}, e^\top p = e^\top d\}.$$

The input vector p must satisfy minimum and maximum generation limits (9e), and the output vector $\tilde{p}$ jointly satisfies minimum/maximum generation limits (9e) and power balance constraint (9b). The projection adjusts all entries in p uniformly until either the total power generation matches the total demand, or the entries in p reach their bounds. Formally, $\tilde{p}$ is obtained as the unique solution of the system of equations

$$\tilde{p} = \text{clamp}(\hat{p}' + \delta, \underline{p}, \bar{p}), \tag{13a}$$

$$e^\top \tilde{p} = e^\top d, \tag{13b}$$

where $\delta \in \mathbb{R}$ is a scalar. Note that (13) can reduced to a uni-dimensional problem in δ by substituting out $\tilde{p}$. Namely, letting $f(\delta) = e^\top \text{clamp}(\hat{p}' + \delta, \bar{p}, \underline{p})$, (13b) reduces to $f(\delta) = e^\top d$. The uniqueness of the solution to (13) then follows from the fact that f is monotonically increasing. δ can be effectively computed using binary search (see Algorithm 2). It can be easily parallelized in PyTorch (Paszke et al., 2019) with its subgradient computed using auto-differentiation.

The Hypersimplex projection layer is then represented as an MILP by encoding the system of equations (13). Namely, constraint (13a) is encoded as an MILP following Equation (11), and constraint (13b) is linear. The compact optimality

Algorithm 2 Hypersimplex Projection via Binary Search

Input: Initial dispatch $\tilde{\boldsymbol{p}}$, Dispatch bounds $\bar{\boldsymbol{p}}$, $\underline{\boldsymbol{p}}$, demand $\boldsymbol{d}$, numerical tolerance ϵ
Initialize $\bar{\delta} = \max(\bar{\boldsymbol{p}} - \underline{\boldsymbol{p}})$, $\underline{\delta} = -\max(\bar{\boldsymbol{p}} - \underline{\boldsymbol{p}})$, $\delta = (\underline{\delta} - \bar{\delta})/2$, $D = \boldsymbol{e}^\top \boldsymbol{d}$
while $|\underline{\delta} - \bar{\delta}| \geq \epsilon$ or $|f(\delta) - D| \geq \epsilon$ **do**
  if $f(\delta) \geq D$ **then**
    $\bar{\delta} = \delta$
  else
    $\underline{\delta} = \delta$
  end if
  $\delta = (\underline{\delta} - \bar{\delta})/2$
end while
return δ

verification problem for DC-OPF proxies thus reads:

$$\max_{\boldsymbol{d} \in \mathcal{X}} \quad \boldsymbol{c}^\top \boldsymbol{p} + M^{\text{th}} \boldsymbol{e}^\top \xi^{\text{th}} - \boldsymbol{c}^\top \tilde{\boldsymbol{p}} - M^{\text{th}} \boldsymbol{e}^\top \tilde{\xi}^{\text{th}} \tag{14a}$$
$$\text{s.t.} \quad \hat{\boldsymbol{p}} = \text{NN}_{\boldsymbol{\theta}}(\boldsymbol{d}), \tag{14b}$$
$$\tilde{\boldsymbol{f}} = \boldsymbol{H}(\tilde{\boldsymbol{p}} - \boldsymbol{d}), \tag{14c}$$
$$\tilde{\xi}^{\text{th}} = \max\{\tilde{\boldsymbol{f}} - \bar{\boldsymbol{f}}, -\bar{\boldsymbol{f}} - \tilde{\boldsymbol{f}}, \boldsymbol{0}\}, \tag{14d}$$
$$(11), (13), \tag{14e}$$
$$(9b) - (9e), \tag{14f}$$
$$\boldsymbol{p} \in \mathbb{R}^B, \hat{\boldsymbol{p}} \in \mathbb{R}^B, \tilde{\boldsymbol{p}} \in \mathbb{R}^B, \xi^{\text{th}} \in \mathbb{R}^E_+, \tilde{\xi}^{\text{th}} \in \mathbb{R}^E_+ \tag{14g}$$

Constraint (14b) encodes the inference of the neural network, which could be linearized by introducing binary decision variables (Tjeng et al., 2019; Bunel et al., 2018). Constraint (14c) and (14d) compute the thermal violation of the predicted dispatch. Before solving formulation (14), all clamp and $\max$ operators are linearized by introducing binary decision variables, which leads to a Mixed Integer Linear Program (MILP).

A.2. Detailed Modeling of Bilevel Formulation

The Bilevel formulation of DC-OPF proxy reads:

$$\max_{\boldsymbol{d} \in \mathcal{X}} \quad \boldsymbol{c}^\top \boldsymbol{p} + M^{\text{th}} \boldsymbol{e}^\top \xi^{\text{th}} - \boldsymbol{c}^\top \tilde{\boldsymbol{p}} - M^{\text{th}} \boldsymbol{e}^\top \tilde{\xi}^{\text{th}} \tag{15a}$$
$$\text{s.t.} \quad (14b) - (14d), (14g) \tag{15b}$$
$$(11), (13), \tag{15c}$$
$$\boldsymbol{p}, \xi^{\text{th}} = \text{DC-OPF}(\boldsymbol{d}) \tag{15d}$$

where lower level DC-OPF($\boldsymbol{d}$) outputs the optimal dispatch and thermal violations by solving formulation (9) with input $\boldsymbol{d}$.

To enhance the tractability, the work in (Nellikkath & Chatzivasileiadis, 2021) reformulates DC-OPF($\boldsymbol{d}$) with KKT conditions:

$$(9b) - (9f) \tag{16a}$$
$$\boldsymbol{e}\lambda + \boldsymbol{H}^\top \underline{\nu} - \boldsymbol{H}^\top \bar{\nu} + \underline{\mu} - \bar{\mu} = c \tag{16b}$$
$$\underline{\nu} + \bar{\nu} + \zeta = M^{\text{th}} \boldsymbol{e} \tag{16c}$$
$$\underline{\mu}, \bar{\mu}, \underline{\nu}, \bar{\nu}, \zeta \geq 0 \tag{16d}$$
$$\underline{\nu}^\top (\boldsymbol{H}\boldsymbol{p} + \xi^{\text{th}} + \bar{\boldsymbol{f}} - \boldsymbol{H}\boldsymbol{d}) = 0 \tag{16e}$$
$$\bar{\nu}^\top (-\boldsymbol{H}\boldsymbol{p} + \xi^{\text{th}} + \bar{\boldsymbol{f}} + \boldsymbol{H}\boldsymbol{d}) = 0 \tag{16f}$$
$$\underline{\mu}^\top (\boldsymbol{p} - \underline{\boldsymbol{p}}) = 0 \tag{16g}$$
$$\bar{\mu}^\top (\bar{\boldsymbol{p}} - \boldsymbol{p}) = 0 \tag{16h}$$
$$\zeta^\top \xi^{\text{th}} = 0 \tag{16i}$$

Constraints (9b) - (9f) model the primal feasibility. Constraints (16b) - (16d) model the dual feasibility, where $\bar{\mu}$ and $\underline{\mu}$ denote the dual variables for the upper and lower bounds in Constraint (9e), respectively. The $\underline{\nu}, \bar{\nu}$ are the dual variables for the upper and lower thermal limits in Constraints (9d) and (9c), respectively. λ denotes the dual variable for the power balance constraint (9b). ζ denotes the dual variable for the non-negativity of thermal violations. Constraints (16e)-(16i) model the complementary slackness, which can be reformulated as an MILP using the standard big-M formulation. Note that valid bounds on all primal variables can be derived from the primal formulation. Dual variables $\underline{\nu}, \bar{\nu}, \zeta$ are naturally bounded by M^{th}. For the remaining dual variables, a large M value is selected.

Finally, the reformulated Bilevel formulation reads

$$\max_{\boldsymbol{d} \in \mathcal{X}} \quad \boldsymbol{c}^\top \boldsymbol{p} + M^{\text{th}} \boldsymbol{e}^\top \xi^{\text{th}} - \boldsymbol{c}^\top \tilde{\boldsymbol{p}} - M^{\text{th}} \boldsymbol{e}^\top \tilde{\xi}^{\text{th}} \tag{17a}$$
$$\text{s.t.} \quad (14b) - (14d), (14g) \tag{17b}$$
$$(11), (13), \tag{17c}$$
$$(9b) - (9f) \tag{17d}$$
$$(16b) - (16i) \tag{17e}$$

A.3. Optimization-based Bound Tightening

The MILP formulations introduce a significant number of binary decision variables. The convergence of these MILP problems becomes notably challenging due to the poor quality of their linear relaxations. Consequently, tightening the bounds of the input variables for activation functions is crucial to enhancing the relaxations and improving the overall solution process. This paper uses Optimization-Based Bound Tightening (OBBT) to refine variable bounds and solution efficiency (Caprara & Locatelli, 2010; Zhao et al., 2024).

Consider a neural network with an input vector $\boldsymbol{x}^{(0)} := \boldsymbol{x}_0 \in \mathbb{R}^{n_0}$. In this network, n_i represents the number of neurons in the i-th layer. The network consists of L layers, where each layer i has an associated weight matrix $\boldsymbol{W}^{(i)} \in \mathbb{R}^{n_i \times n_{i-1}}$ and a bias vector $\boldsymbol{b}^{(i)} \in \mathbb{R}^{n_i}$, for $i \in \{1, \ldots, L\}$. Let $\boldsymbol{y}^{(i)}$ denote the pre-activation vector and $\boldsymbol{x}^{(i)}$ the post-activation vector at layer i, with $\boldsymbol{x}^{(i)} = \sigma(\boldsymbol{y}^{(i)})$ where σ could be any activation function. Define f to be the desired property for any input $\boldsymbol{x}^{(0)} \in \mathcal{C}$.

The optimization problem is as follows:

$$\begin{aligned} \min \quad & f(\boldsymbol{y}^{(L)}) \\ \text{s.t.} \quad & \boldsymbol{y}^{(i)} = \boldsymbol{W}^{(i)} \boldsymbol{x}^{(i-1)} + \boldsymbol{b}^{(i)}, && \forall i \in \{1, \ldots, L\}, \\ & \boldsymbol{x}^{(i)} = \sigma(\boldsymbol{y}^{(i)}), && \forall i \in \{1, \ldots, L-1\}, \\ & \boldsymbol{x}^{(0)} \in \mathcal{C}. \end{aligned} \tag{18}$$

To tighten variable bounds, OBBT solves two optimization subproblems for each neuron to determine its maximum and minimum bounds (Caprara & Locatelli, 2010; Zhao et al., 2024). Specifically, let $\boldsymbol{y}_k^{(t)}$ denote the k-th neuron at layer t

subject to network constraints. Given $0 \leq t \leq L$, the problem states:

$$\begin{aligned}
\text{max/min} \quad & \boldsymbol{y}_k^{(t)} \\
\text{s.t.} \quad & \boldsymbol{y}^{(i)} = \boldsymbol{W}^{(i)}\boldsymbol{x}^{(i-1)} + b^{(i)}, && \forall i \in \{1,\dots,t\}, \\
& \boldsymbol{x}^{(i)} = \sigma(\boldsymbol{y}^{(i)}), && \forall i \in \{1,\dots,t-1\}, \\
& \boldsymbol{x}^{(0)} \in \mathcal{C}, \\
& \boldsymbol{y}_l^{(i)} \leq \boldsymbol{y}^{(i)} \leq \boldsymbol{y}_u^{(i)}, && \forall i \in \{1,\dots,t\}, \\
& \boldsymbol{x}_l^{(i)} \leq \boldsymbol{x}^{(i)} \leq \boldsymbol{x}_u^{(i)} && \forall i \in \{1,\dots,t-1\}.
\end{aligned} \tag{19}$$

A.4. More results on optimality verification of DCOPF proxies

Table 7 reports the final objective and solving time on small power systems, where Compact formulation can solve all instances to optimality within 1 minute.

Table 7. Comparisons of PGA-VFA and Compact formulation on final objective and solving time on small power systems. T2PGA reports the time that the compact formulation takes to reach the same primal objective of the proposed PGA-VFA. Compact formulation is very effective at finding optimal solutions for small systems i.e., it solves all instances to optimality within 1 minute.

		PGA-VFA		Compact		
System	$\%u$	ObjVal ($)	Time (s)	ObjVal ($)	Time (s)	T2PGA (s)
57	0	127.36	2.17	141.36	0.31	0.10
	1	137.35	1.44	142.29	0.59	0.11
	2	143.70	4.31	151.54	0.54	0.12
	5	171.53	3.35	179.12	1.10	1.10
	10	175.89	2.28	432.20	1.21	1.10
	20	1527.72	5.59	3837.33	0.54	0.30
118	0	752.29	1.43	6216.97	1.20	0.74
	1	4615.37	6.96	7917.56	2.07	1.17
	2	4615.37	7.61	9607.68	3.08	2.00
	5	4615.37	4.44	14329.88	12.08	3.00
	10	5448.69	6.26	68556.32	14.14	6.67
	20	6048.77	9.04	69109.46	59.41	17.27

B. More information of Knapsack

Recall the knapsack problem formulation

$$\max_{\boldsymbol{y}} \quad \boldsymbol{v}^T\boldsymbol{y} \tag{20a}$$

$$s.t. \quad \boldsymbol{w}^T\boldsymbol{y} \leq l, \tag{20b}$$

$$\boldsymbol{y} \in \{0,1\}^K. \tag{20c}$$

B.1. Detailed Modeling of Compact Formulation

Figure 7 illustrates the Knapsack proxies. First, a fully-connected neural network with ReLU activation predicts a score for each item, where a higher score indicating higher desirability of the item. Then, a feasibility repair step sorts the items by the predicted score in a descending order and adds items to the knapsack following the order until reaching the knapsack's capacity limit, addressing constraint (20b). The MLP is trained disjointly with the feasibility repair steps in the training due to their non-differentiability. The neural network is trained using supervised learning to minimize the distances of the predicted scores to the ground truth: $l(\boldsymbol{s}, \boldsymbol{s}^*) = \|\boldsymbol{s} - \boldsymbol{s}^*\|_2$. The ground truth score is computed heuristically: $\boldsymbol{s}^* = \frac{\boldsymbol{v}}{\boldsymbol{w}}$, where $\boldsymbol{v}$ denotes the value of the items and $\mathbf{w}$ denotes the weight of the items. Note that the score is exactly the solution of the continuous relaxation of Knapsack problems.

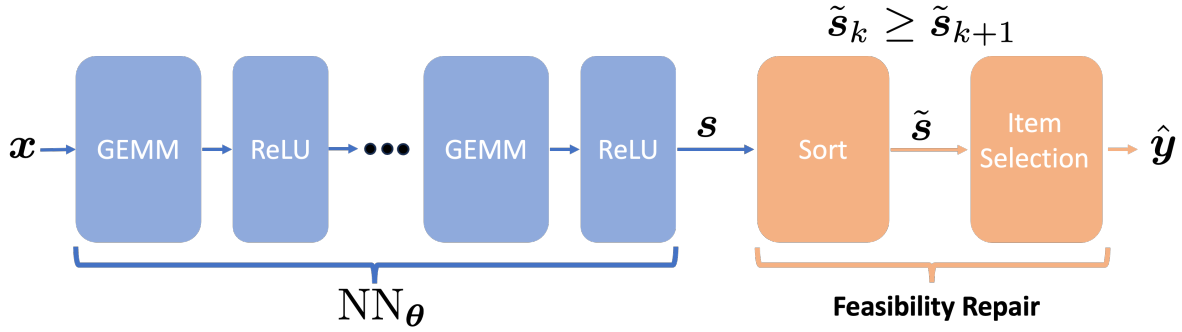


Figure 7. Optimization Proxies for Knapsack

The following part focuses on formulating the sorting and item selection operators into MILP.

Sorting Denote the predicted score $\boldsymbol{s}$ from the neural network, the sorting operation sorts the item id with the score descendingly. The sorting operation is modeled by introducing a permutation matrix:

$$\tilde{\boldsymbol{s}} = \boldsymbol{p}\boldsymbol{s}, \tag{21a}$$
$$\tilde{\boldsymbol{s}}_k \geq \tilde{\boldsymbol{s}}_{k+1}, \forall k \in [K] \tag{21b}$$
$$\sum_{i}^{K} \boldsymbol{p}_{i,k} = 1, \forall k \in [K], \tag{21c}$$
$$\sum_{k}^{K} \boldsymbol{p}_{i,k} = 1, \forall i \in [K], \tag{21d}$$
$$\boldsymbol{p} \in \{0,1\}^{K \times K}, \tag{21e}$$
$$\boldsymbol{s} \in \mathbb{R}_+^K, \tilde{\boldsymbol{s}} \in \mathbb{R}_+^K. \tag{21f}$$

The $\tilde{\boldsymbol{s}}$ denotes the score after the sortation. The $\boldsymbol{p}$ denotes the permutation matrix to convert predicted score $\boldsymbol{s}$ to sorted score $\tilde{\boldsymbol{s}}$. Constraint (21c)-(21e) define the permutation matrix and constraint (21a) and (21b) ensure the perturbation matrix encodes the orders of sorting the score descendingly.

Item Selection After obtaining the permutation matrix, the item selection operator could be modeled as:

$$\tilde{\boldsymbol{y}} = \boldsymbol{p}\hat{\boldsymbol{y}}, \tag{22a}$$
$$\tilde{\boldsymbol{w}} = \boldsymbol{p}\boldsymbol{w}, \tag{22b}$$
$$\boldsymbol{w}^T \hat{\boldsymbol{y}} \leq l, \tag{22c}$$
$$\tilde{\boldsymbol{y}}_k \geq \tilde{\boldsymbol{y}}_{k+1}, \forall k \in [K] \tag{22d}$$
$$\sum_{i=1}^{k} \tilde{\boldsymbol{w}}_i \geq (1 - \tilde{\boldsymbol{y}}_k)(l+1), \forall k \in [K] \tag{22e}$$
$$\tilde{\boldsymbol{y}} \in \{0,1\}^K, \hat{\boldsymbol{y}} \in \{0,1\}^K. \tag{22f}$$

Constraint (22a) and (22b) permutate the item action and weights. Constraint (22c) ensures the total weight of the added items is less than the knapsack capacity. Constraint (22d) ensures items are added sequentially following the sorted scores Constraint (22e) ensures that the addition of items continues only if adding another item does not exceed the knapsack's capacity. Finally, following the reformulation of the operations and Formulation (5), the compact optimality verification for Knapsack proxies could be written as a MILP and readily be solved by solvers like Gurobi (Gurobi Optimization, LLC, 2023).

Finally, the optimality verification formulation reads:

$$\max_{\boldsymbol{x}=\{\boldsymbol{v},l\}\in\mathcal{X}} \quad \boldsymbol{v}^T\boldsymbol{y} - \boldsymbol{v}^T\hat{\boldsymbol{y}} \tag{23a}$$

$$s.t. \quad \boldsymbol{s} = \text{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{23b}$$

$$(21), (22), \tag{23c}$$

$$(20b), (20c) \tag{23d}$$

The objective function (23a), constraints (23b) and (23c) are linearized by introducing binary decision variables, which gives a MILP.