CONTROLLING SCALE MISMATCH IN DECISION-FOCUSED LEARNING VIA COST REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision-focused learning (DFL) trains predictive models to minimize the cost of the decisions they induce rather than prediction error. When the model predicts the objective coefficients of a combinatorial optimization problem, the map from costs to decisions is piecewise constant, and many state-of-the-art methods obtain informative gradients by perturbing the predicted costs. We show that these methods share a common weakness: the perturbation scale is fixed, but the scale of the predicted costs is not. Because decisions are unchanged when costs are rescaled, the training loss does not constrain that scale, and it can drift freely during training. When predicted costs grow too large relative to the perturbations, the perturbed decisions stop differing from the unperturbed ones, and learning stalls or reduces to imitation. When they become too small, the perturbations dominate and the gradients no longer reflect the prediction. Since this drift varies across samples and over training, tuning the perturbation scale alone cannot prevent it. We propose a simple, method-agnostic remedy: a smooth shrinkage of the predicted costs, which leaves decisions unchanged and adds negligible cost. Experiments with three DFL methods show that it markedly reduces regret on a problem designed to induce this instability, and on standard benchmarks it corrects severe training failures at little cost when none occur.

1 INTRODUCTION

We focus on the problem of estimating objective function costs in combinatorial optimization problems formulated as *Mixed-Integer Linear Programs* (MILP). *Decision-Focused Learning* (DFL) has emerged as a promising framework for this task, as it integrates the optimization problem directly into the learning process, thereby improving predictive performance with respect to the downstream decision-making objective. However, the inherently piecewise-constant nature of the mapping from predicted costs to optimal decisions poses a significant challenge for gradient-based learning. To address this issue, numerous approximation methods have been proposed in recent years, enabling learning through discrete optimization models. Most of these approaches rely on perturbing the cost vector to infer informative descent directions. In practice, however, selecting and tuning the magnitude of these perturbations is often difficult and can lead to suboptimal performance.

Our primary contribution is to expose a limitation shared by all examined perturbation-based DFL techniques, grounded in key principles of optimal solution stability in combinatorial optimization. Specifically, we demonstrate that shifts in the relative scale between perturbations and cost vectors during the learning phase can negatively impact performance. Our study shows that regularizing the estimated costs provides better control over the decision-focused learning process, a finding supported by numerical experiments over multiple decision-focused learning methods and optimization problems.

2 RELATED WORK AND CONTRIBUTIONS

The integration of prediction and optimization in ML models has gained significant attention in recent years for its potential to improve decision-making under uncertainty (Kotary et al., 2020; Mandi et al., 2024).

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

Our study focuses on DFL techniques that leverage additive cost estimate perturbations for optimization mapping differentiation : such methods are prevalent in the literature, although multiple approaches exist for learning the cost objective of mixed-integer linear programs that do not rely on perturbation-based gradient estimation (see e.g. Mandi et al. (2024)). All methods under study address empirical cost minimization: given a historical dataset of feature and cost vector pairs, they learn from experience the cost estimate that best supports the decision to be made. Originating from implicit differentiation via parameter perturbations (Domke, 2010), this idea was adapted for DFL by considering distributions over decisions rather than single optimal solutions, enabling differentiation through discrete mappings. This adaptation produced the *Perturb-and-MAP* framework (Papandreou & Yuille, 2011), later refined in Implicit Maximum Likelihood Estimation (IMLE) (Niepert et al., 2021), which improved perturbation sampling. The *Differentiable Black-Box* (DBB) approach (Pogancic et al., 2021) built on similar principles but employed different perturbation strategies, while *Differentiable Perturbed Optimizers* (DPO) (Berthet et al., 2020) extended Perturb-and-MAP by treating the optimization mapping as a deterministic function of both cost estimates and perturbations. Lastly, we take an interest in the *Smart Predict-then-Optimize* (SPO) loss (Elmachtoub & Grigas, 2022) : it corresponds to a surrogate loss function whose sub-gradients indicate worthwhile descent directions. We outline that it relies on the introduction of a perturbation for optimization mapping differentiation, hence the need for regularization.

Several studies on decision-focused learning mention regularization or normalization of cost vector estimates, but consistently as a problem-specific adjustment rather than a general improvement to state-of-the-art approaches. Sahoo et al. (2022), Schutte et al. (2026), and Wan & Liu (2026) employ it to address solution invariance in the design of their respective methods. Our contribution is of a different nature: rather than treating this as a static design question, we identify a dynamic one. Because the cost vector estimates drift over the course of training, the effect of the perturbation on the perturbation-based DFL methods under study changes as learning proceeds and may lead to overall training collapse. We address this drift with a method-agnostic and problem-agnostic scheme that is simple to implement.

Finally, Aubin-Frankowski et al. (2026) derive generalization bounds for perturbation-smoothed linear oracles via a fan-crossing probability closely related to the notion of solution stability, though without addressing the specific DFL losses or training collapse studied here. Our results do not rely on theirs, but their bound corroborates our observations.

3 General Framework

We first introduce in this section all the elements necessary to establish our core contribution, that is determining general efficient methods to avoid training collapse in perturbation-based DFL due to solution stability and solution invariance.

Consider a convex cone $\Theta \subset \mathbb{R}^n$ and a non-empty finite set of distinct points $\mathbb{Y} \subset \mathbb{R}^n$ with $\mathcal{C}$ its convex hull. We consider a general MILP minimization problem with linear costs parameterized by a vector $\boldsymbol{\theta} \in \Theta$ and an associated *optimization mapping* $f : \Theta \to \mathcal{C}$ defined as follows:

$$\min_{y \in \mathcal{C}} \boldsymbol{\theta}^\intercal y, \quad f(\boldsymbol{\theta}) = \arg\min_{y \in \mathcal{C}} \boldsymbol{\theta}^\intercal y \tag{1}$$

Function f maps a cost vector $\boldsymbol{\theta} \in \Theta$ to an optimal decision $f(\boldsymbol{\theta}) \in \mathcal{C}$. Our goal is to estimate a cost vector $\boldsymbol{\theta}$ whose quality depends on the resulting decision $f(\boldsymbol{\theta})$.

3.1 Learning from Decisions

We design a machine-learning model parameterized by $\boldsymbol{v}$ and described by equations:

$$\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \quad \boldsymbol{y} = f(\boldsymbol{\theta}) \tag{2}$$

Vector $\boldsymbol{x} \in \mathbb{X}$ denotes feature inputs while $\boldsymbol{y} \in \mathbb{Y}$ is a target output representing a solution to a discrete decision model parameterized by $\boldsymbol{\theta} \in \Theta$. Function $h_{\boldsymbol{v}} : \mathbb{X} \to \Theta$ is a differentiable parameterized mapping while $f : \Theta \to \mathbb{Y}$ represents an optimization mapping as described above. Here, the optimization mapping can be seen as a final combinatorial optimization layer within our ML model. We consider a training loss function based on an evaluation $l(\boldsymbol{y}, \bar{\boldsymbol{\theta}})$ of the estimated

decision $\boldsymbol{y} = f(h_{\boldsymbol{v}}(\boldsymbol{x}))$ with respect to the ground-truth cost vector $\bar{\boldsymbol{\theta}}$. The loss is defined as:

$$L(\boldsymbol{x}, \bar{\boldsymbol{\theta}}; \boldsymbol{v}) = l(\boldsymbol{y}, \bar{\boldsymbol{\theta}}), \quad \text{with } \boldsymbol{y} = f(h_{\boldsymbol{v}}(\boldsymbol{x})) \tag{3}$$

This DFL approach aims to predict the cost objective associated with the optimization mapping. Given a dataset $\{(\boldsymbol{x}_j, \bar{\boldsymbol{\theta}}_j)\}_{j=1}^{N}$, we learn parameters $\boldsymbol{v}$ of equation 2 to minimize the average loss $L(\boldsymbol{x}_j, \bar{\boldsymbol{\theta}}_j; \boldsymbol{v})$ across all samples. In other words, we aim at learning a model that minimizes the evaluation value of decisions $\boldsymbol{y} = f(\boldsymbol{\theta})$ made based on cost vector estimates $\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x})$.

We focus in this work on the *regret* loss L^r, which quantifies how much worse the decision based on predicted parameters is compared to the decision based on the true parameters. Specifically, it evaluates the decision estimate $f(\boldsymbol{\theta}) \in \mathcal{C} \subset \mathbb{R}^n$ with respect to the ground-truth cost vector $\bar{\boldsymbol{\theta}} \in \Theta \subset \mathbb{R}^n$.

$$L^r(\boldsymbol{x}, \bar{\boldsymbol{\theta}}; \boldsymbol{v}) = \bar{\boldsymbol{\theta}}^\intercal \boldsymbol{y} - \bar{\boldsymbol{\theta}}^\intercal f(\bar{\boldsymbol{\theta}}), \quad \text{with } \boldsymbol{y} = f(h_{\boldsymbol{v}}(\boldsymbol{x})) \tag{4}$$

Although our examples and numerical experiments focus on this specific loss function and model structure, our contributions generalize to all studied perturbation-based DFL techniques within the broader framework introduced in equation 2.

Differentiating L^r with regard to $\boldsymbol{v}$ is necessary in order to train our model. By chain rule, it is therefore necessary to differentiate L^r with regard to $\boldsymbol{\theta}$; its gradient is expressed as follows:

$$\nabla_{\boldsymbol{\theta}} L^r(\boldsymbol{x}, \bar{\boldsymbol{\theta}}; \boldsymbol{v}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})^\intercal \nabla_{\boldsymbol{y}} L^r(\boldsymbol{x}, \bar{\boldsymbol{\theta}}; \boldsymbol{v}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})^\intercal \bar{\boldsymbol{\theta}} \tag{5}$$

Differentiating L^r is challenging mainly because $\nabla_{\boldsymbol{\theta}} L^r(\boldsymbol{x}, \bar{\boldsymbol{\theta}}; \boldsymbol{v})$ depends on $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$. The optimization mapping f is typically assumed to be piecewise constant, so the gradient $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ is zero almost everywhere. Since the gradient provides little value for optimization, DFL methods are applied to find effective descent directions for minimizing the loss function. This is typically achieved by introducing cost perturbations within the optimization mapping, hence the prevalence of perturbation-based techniques among state-of-the-art DFL approaches.

3.2 Learning from Decisions by Imitation or Experience

DFL techniques are classified by Bengio et al. (2021) based on whether they learn by imitation or by experience. In the imitation setting, optimal decisions are inferred by replicating expert-provided behavior, without direct optimization of a performance measure. In contrast, experience-based learning involves iterative trial-and-error, guided by rewards and penalties, with the explicit goal of optimizing a defined performance metric.

Our objective is to demonstrate that, without controlling solution stability, DFL techniques designed for learning through experience become either inefficient or degenerate into imitation-based learning. We regard both outcomes as undesirable. Collapsing to imitation discards the experience-based signal altogether, and the cost of doing so is problem-dependent: it may be benign on some instances and severely detrimental on others. To that end, we study Fenchel-Young (FY) losses (Blondel et al., 2020). In particular, Fenchel-Young losses have been used as an imitation-based loss in decision-focused learning when the target is a ground-truth decision (Parmentier, 2025; Dalle et al., 2022), with gradient:

$$\nabla_{\boldsymbol{\theta}} L^{FY}(\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}) = f(\bar{\boldsymbol{\theta}}) - \tilde{f}(\boldsymbol{\theta}). \tag{6}$$

A descent direction toward a minimum is derived from gradient $\nabla_{\boldsymbol{\theta}} L^{FY}(\boldsymbol{\theta}, \bar{\boldsymbol{\theta}})$. Following this direction produces cost vector estimates whose optimization mapping best replicates the ground-truth decision $f(\bar{\boldsymbol{\theta}})$. We stress that this imitation behavior comes from the choice of target $f(\bar{\boldsymbol{\theta}})$.

In the following we show that under certain conditions, the gradient estimates of several perturbation-based DFL techniques can resemble this particular gradient equation 6, effectively categorizing them as learning processes by imitation.

3.3 Solution Stability and Solution Invariance

It is commonly assumed that the optimization mapping f is piecewise constant and thus possesses a gradient that is of little relevance in an optimization process (Mandi et al., 2024). We strengthen this claim by reframing it into a stronger formalism based on combinatorial optimization literature related to optimal solution stability (Bonnans & Shapiro, 2000). Moreover, we highlight how solution

stability can characterize the impact of additive cost perturbations on the optimization mapping. The presented results are illustrated by an example in appendices A.1.

Let us denote by F a set valued optimization mapping such that $F(\boldsymbol{\theta})$ represents the exhaustive set of optimal solutions for the problem defined by equation 1 with cost vector $\boldsymbol{\theta} \in \Theta$. In other words, $F(\boldsymbol{\theta})$ represents the set of all possible values taken by $f(\boldsymbol{\theta})$. In most cases, the optimal solution set $F(\boldsymbol{\theta})$ is the singleton $f(\boldsymbol{\theta})$ (Elmachtoub & Grigas, 2022). Then by upper semi-continuity of F (Bonnans & Shapiro, 2000) there exists a supremum $\eta(\boldsymbol{\theta})$ called *stability radius* such that every perturbed cost vector $\tilde{\boldsymbol{\theta}} \in \Theta$ lying within distance $\eta(\boldsymbol{\theta})$ of $\boldsymbol{\theta}$ (in its *stability region*) shares the same optimal decision set as $\boldsymbol{\theta}$, that is, $F(\tilde{\boldsymbol{\theta}}) = F(\boldsymbol{\theta})$ (Sotskov et al., 1995). We conclude that, in most cases, the set valued optimization mapping F is constant within a stability region around $\boldsymbol{\theta}$, depending on a stability radius η.

For all $\boldsymbol{\theta} \in \Theta$, let $\mathcal{P}(\boldsymbol{\theta})$ denote the set of perturbations $\boldsymbol{\delta}$ such that $\boldsymbol{\theta} + \boldsymbol{\delta} \in \Theta$. We deduce from the previous developments the following property.

Property 1. *Consider an optimization mapping $f : \Theta \to \mathcal{C}$, as well as a cost vector $\boldsymbol{\theta}$ and a perturbation $\boldsymbol{\delta} \in \mathcal{P}(\boldsymbol{\theta})$. If $\boldsymbol{\theta} + \boldsymbol{\delta}$ is in the stability region of $\boldsymbol{\theta}$ and $F(\boldsymbol{\theta})$ is the singleton $f(\boldsymbol{\theta})$ then $f(\boldsymbol{\theta} + \boldsymbol{\delta}) = f(\boldsymbol{\theta})$. If $\boldsymbol{\theta} + \boldsymbol{\delta}$ in the stability region of $\boldsymbol{\delta}$, one has $\boldsymbol{\delta} \in \Theta$ and $F(\boldsymbol{\delta})$ is the singleton $f(\boldsymbol{\delta})$ then $f(\boldsymbol{\theta} + \boldsymbol{\delta}) = f(\boldsymbol{\delta})$.*

The stability property 1 represents the main motivation for an efficient management of solution stability in decision-focused learning, due to its impact combined with *solution invariance* (see e.g. Sahoo et al. (2022)) which can be defined as follows:

Property 2. *Let us consider an optimization mapping $f : \Theta \to \mathcal{C}$. For all cost vectors $\boldsymbol{\theta} \in \Theta$ and positive scalar $\gamma > 0$, one has $f(\gamma \cdot \boldsymbol{\theta}) = f(\boldsymbol{\theta})$.*

We examine in the following section how solution stability together with solution invariance can negatively impact the perturbation-based decision-focused learning process.

4 Impacts of Solution Stability on Perturbation-Based DFL Techniques

4.1 Detrimental Effects of Solution Stability on the Decision-Focused Learning Process

Due to the invariance property 2, we can analytically express the stability radius of a cost vector according to its norm and the stability radius of its directional component.

Proposition 4.1. *Consider a non-constant optimization mapping $f : \Theta \to \mathcal{C}$ and a cost vector $\boldsymbol{\theta} \in \Theta$ such that $F(\boldsymbol{\theta})$ is the singleton $f(\boldsymbol{\theta})$. The stability radius of $\boldsymbol{\theta}$ is proportional to its norm:*

$$\eta(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\| \cdot \eta\left(\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}\right) \tag{7}$$

With $\eta\left(\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}\right) \in (0,\, 1]$.

A detailed proof of this proposition is provided in appendices A.2. This result on the stability radius is key to understanding why cost vector estimates must be regularized in perturbation-based decision-focused learning. Indeed, methods in this class rely on a perturbed decision mapping $\tilde{f}\colon \Theta \to \mathcal{C}$, which adds a random perturbation to the input cost vector before the corresponding optimization mapping is evaluated. In other words for a cost vector $\boldsymbol{\theta} \in \Theta$ and a cost perturbation $\boldsymbol{\delta} \in \mathcal{P}(\boldsymbol{\theta})$, the corresponding perturbed decision is:

$$\tilde{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta} + \boldsymbol{\delta}) \tag{8}$$

DFL techniques leverage the perturbed optimization mapping to identify relevant descent directions minimizing the loss function by exploration in the neighborhood of $f(\boldsymbol{\theta})$ and comparison of perturbed decisions $\tilde{f}(\boldsymbol{\theta})$ with $f(\boldsymbol{\theta})$. Hence, it is necessary that all perturbed decisions remain close to but different from $f(\boldsymbol{\theta})$. However, Proposition 4.1 outlines that it may not be the case in presence of a scale mismatch between the cost vector $\boldsymbol{\theta}$ and the perturbation $\boldsymbol{\delta}$. Let us define the following ratio:

Definition 4.2. Given a cost vector $\boldsymbol{\theta} \in \Theta$ and a perturbation $\boldsymbol{\delta} \in \mathcal{P}(\boldsymbol{\theta})$, we define the cost vector stability ratio as $\rho_\eta(\boldsymbol{\theta}, \boldsymbol{\delta}) = \|\boldsymbol{\delta}\| \cdot \eta(\boldsymbol{\theta})^{-1}$.

If $\rho_\eta(\boldsymbol{\theta}, \boldsymbol{\delta}) < 1$, the perturbed decision $f(\boldsymbol{\theta} + \boldsymbol{\delta})$ is equal to the exact decision $f(\boldsymbol{\theta})$. In this situation, the neighborhood around $f(\boldsymbol{\theta})$ is not effectively explored, making it hardly possible to identify a relevant descent direction. Let us now define the following ratio:

Definition 4.3. Given a cost vector $\boldsymbol{\theta} \in \Theta$ and a perturbation $\boldsymbol{\delta} \in \mathcal{P}(\boldsymbol{\theta})$, we define the perturbation stability ratio as $\xi_\eta(\boldsymbol{\theta}, \boldsymbol{\delta}) = \|\boldsymbol{\theta}\| \cdot \eta(\boldsymbol{\delta})^{-1}$.

If $\xi_\eta(\boldsymbol{\theta}, \boldsymbol{\delta}) < 1$, the perturbed decision $f(\boldsymbol{\theta} + \boldsymbol{\delta})$ is equal to $f(\boldsymbol{\delta})$. Because $f(\boldsymbol{\delta})$ generally lies far from the neighborhood of $f(\boldsymbol{\theta})$, the resulting descent direction may be misleading.

Leaving the stability ratios ρ_η and ξ_η uncontrolled during training can prevent the optimization mapping from differentiating effectively with regard to cost vectors, which ultimately degrades performance. Proposition 4.1 establishes that, for any cost vector, the stability radius is proportional to its norm. Hence cost vector estimates and perturbations should remain on the same scale throughout training so that both ratios remain above one.

In the remainder of this section, we characterize how poorly managed stability ratios affect each state-of-the-art perturbation-based DFL method. Building on this analysis, we then introduce regularization approaches that address the issues we identify.

4.2 DFL Degenerating into Imitation or Collapse

Smart-Predict-then-Optimize The Smart-Predict-then-Optimize (SPO) loss serves as a differentiable surrogate for the regret loss in equation 4. Hence its purpose is to learn from decisions by experience to minimize regret. Its gradient is approximated by $\nabla_{\boldsymbol{\theta}} L_\alpha^{SPO}(\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}) \approx f(\bar{\boldsymbol{\theta}}) - f(\alpha\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})$, with $\alpha > 0$.

If $\rho_\eta(\alpha\boldsymbol{\theta}, -\bar{\boldsymbol{\theta}}) < 1$, then $f(\alpha\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) = f(\boldsymbol{\theta})$, so $\nabla_{\boldsymbol{\theta}} L_\alpha^{SPO} \approx f(\bar{\boldsymbol{\theta}}) - f(\boldsymbol{\theta})$. Hence the gradient resembles that of equation 6 and the learning process turns into imitation.

If $\xi_\eta(\alpha\boldsymbol{\theta}, -\bar{\boldsymbol{\theta}}) < 1$, then $f(\alpha\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) = f(-\bar{\boldsymbol{\theta}})$, yielding: $\nabla_{\boldsymbol{\theta}} L_\alpha^{SPO} \approx f(\bar{\boldsymbol{\theta}}) - f(-\bar{\boldsymbol{\theta}})$. Hence information about $\boldsymbol{\theta}$ is discarded, making differentiation ineffective.

Thus, training with SPO loss may shift from experience-based learning to imitation or collapse depending on the relative scales of $\alpha\boldsymbol{\theta}$ and $\bar{\boldsymbol{\theta}}$.

Implicit differentiation by perturbation We examine DFL techniques that approximate optimization mappings via a probabilistic perspective, including Perturb-and-MAP, Implicit MLE (IMLE) and Differentiable BlackBox (DBB). These methods are compatible with any differentiable loss function, making them inherently experience-based since they allow models to be trained according to arbitrary performance metrics.

Let us define a distribution of decisions $p(\boldsymbol{y}|\boldsymbol{\theta})$ over $\mathcal{C}$ and approximate $f(\boldsymbol{\theta})$ by a perturbed optimizer $\mathbb{E}_p[\boldsymbol{y}|\boldsymbol{\theta}]$. Differentiation relies on implicit perturbation (Domke, 2010), yielding for the regret loss L^r:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \bar{\boldsymbol{\theta}}; \boldsymbol{v}) \approx \beta^{-1} \cdot \left(\mathbb{E}_p[\boldsymbol{y}|\boldsymbol{\theta} + \beta \cdot \bar{\boldsymbol{\theta}}] - \mathbb{E}_p[\boldsymbol{y}|\boldsymbol{\theta}]\right). \tag{9}$$

If $\rho_\eta(\boldsymbol{\theta}, \beta\bar{\boldsymbol{\theta}}) < 1$, then: $\nabla_{\boldsymbol{\theta}} L \approx \beta^{-1} \cdot (\mathbb{E}_p[\boldsymbol{y}|\boldsymbol{\theta}] - \mathbb{E}_p[\boldsymbol{y}|\boldsymbol{\theta}]) = 0$. Hence no descent direction is found.

If $\xi_\eta(\boldsymbol{\theta}, \beta\bar{\boldsymbol{\theta}}) < 1$, then $\nabla_{\boldsymbol{\theta}} L \approx \beta^{-1} \cdot (\mathbb{E}_p[\boldsymbol{y}|\bar{\boldsymbol{\theta}}] - \mathbb{E}_p[\boldsymbol{y}|\boldsymbol{\theta}])$. Here the gradient reduces to $\nabla_{\boldsymbol{\theta}} L^{FY}(\boldsymbol{\theta}, \bar{\boldsymbol{\theta}})$ corresponding to that of equation 6 and the learning process degenerates to imitation.

We conclude that perturbations introduced by $\beta\bar{\boldsymbol{\theta}}$ must remain on the same scale as $\boldsymbol{\theta}$ to prevent the learning process from collapsing or degenerating into imitation.

Differentiable Perturbed Optimizers Differentiable Perturbed Optimizers (DPO) adopt a blackbox approach similar to implicit differentiation by perturbations, exploring the neighborhood of $f(\boldsymbol{\theta})$ through additive perturbations. DPO is a differentiation mechanism compatible with either imitation-based or experience-based learning regimes ; we focus on the second variant. Its perturbed optimizer is defined as: $\tilde{f}_\epsilon(\boldsymbol{\theta}) = \mathbb{E}[f(\boldsymbol{\theta} + \epsilon\mathbf{d})]$.

Here, $\mathbf{d}$ is a random vector whose distribution has a positive differentiable density $p(\mathbf{d}) \propto \exp(-\nu(\mathbf{d}))$, where ν denotes the associated negative log-density (Berthet et al., 2020). The perturbed optimizer is differentiated as follows:

$$\nabla_{\boldsymbol{\theta}} \tilde{f}_{\epsilon}(\boldsymbol{\theta}) = \epsilon^{-1} \cdot \mathbb{E}[f(\boldsymbol{\theta} + \epsilon\mathbf{d}) \cdot \nabla_{\mathbf{d}} \nu(\mathbf{d})^{\intercal}]. \tag{10}$$

If $\rho_{\eta}(\boldsymbol{\theta}, \epsilon\mathbf{d}) < 1$, then $f(\boldsymbol{\theta} + \epsilon\mathbf{d}) \approx f(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} \tilde{f}_{\epsilon}(\boldsymbol{\theta}) \approx 0$, making differentiation ineffective.

If $\xi_{\eta}(\boldsymbol{\theta}, \epsilon\mathbf{d}) < 1$, then $f(\boldsymbol{\theta} + \epsilon\mathbf{d}) \approx f(\mathbf{d})$, so: $\nabla_{\boldsymbol{\theta}} \tilde{f}_{\epsilon}(\boldsymbol{\theta}) = \epsilon^{-1} \cdot \mathbb{E}[f(\mathbf{d}) \cdot \nabla_{\mathbf{d}} \nu(\mathbf{d})^{\intercal}]$. Hence all information about $\boldsymbol{\theta}$ is lost, preventing the identification of meaningful descent directions.

Thus, perturbations $\epsilon\mathbf{d}$ must remain at scale with $\boldsymbol{\theta}$ for efficient differentiation.

4.3 Impact of Stability Ratios in Practice

The two failure modes are not equally hard to control. When $\xi_{\eta}(\boldsymbol{\theta}, \boldsymbol{\delta}) < 1$, the gradient is a nonzero difference of decisions in $\mathcal{C}$ that stays roughly the same across steps, so updates tend to increase $\|\boldsymbol{\theta}\|$ until the ratio exceeds one. If this self-correction fails, the stability radius of $\boldsymbol{\delta}$ must greatly exceed $\|\boldsymbol{\theta}\|$. By Proposition 4.1, this radius scales with $\|\boldsymbol{\delta}\|$, which depends on hyperparameters, so careful tuning is sufficient.

The case $\rho_{\eta}(\boldsymbol{\theta}, \boldsymbol{\delta}) < 1$ is more critical: the gradient vanishes for implicit differentiation and perturbed optimizers, and SPO degenerates into imitation with no guarantee of recovery. Tuning cannot sufficiently prevent this. Because of solution invariance, the regret loss does not depend on $\|\boldsymbol{\theta}\|$, so nothing stops the norm from growing during training. By Proposition 4.1, $\eta(\boldsymbol{\theta})$ then grows at the same rate and ρ_{η} shrinks in inverse proportion. Keeping ρ_{η} above one therefore requires acting dynamically on the cost vector estimate itself during training, which is what Section 5 proposes.

5 Steering the Cost Vector Stability Radius with Cost Regularization

Steering the cost vector stability radius poses a significant challenge since the stability radius is not a fixed, problem-independent constant. Aubin-Frankowski et al. (2026) show that controlling it requires additional structural assumptions rather than a single instance-independent definition. This corroborates our position: the stability radius is inherently a function of the combinatorial geometry of the problem at hand rather than a quantity we could express in closed form.

By Proposition 4.1, the stability radius $\eta(\boldsymbol{\theta})$ of a cost vector $\boldsymbol{\theta}$ is upper-bounded by its norm $\|\boldsymbol{\theta}\|$. Bounding $\|\boldsymbol{\theta}\|$ thus upper-bounds $\eta(\boldsymbol{\theta})$, which in turn lower-bounds $\rho_{\eta}(\boldsymbol{\theta}, \cdot)$ and helps keep it above one. We therefore propose to enforce an upper bound on $\|\boldsymbol{\theta}\|$ through a regularization function applied to the cost vector estimate $\boldsymbol{\theta}$ before it is perturbed.

Consider a perturbed optimization mapping $\tilde{f} : \Theta \to \mathcal{C}$ as defined in equation 8. The regularization function is defined as a mapping $r : \Theta \to \Theta$ satisfying $f(r(\boldsymbol{\theta})) = f(\boldsymbol{\theta})$, so that the exact decision mapping remains unchanged. The role of r is to rescale $\boldsymbol{\theta}$ to match the norm of the perturbation $\boldsymbol{\delta}$ introduced by the perturbed optimization mapping $\tilde{f}$, such that $\tilde{f}(r(\boldsymbol{\theta})) = f(r(\boldsymbol{\theta}) + \boldsymbol{\delta})$.

The regularized decision-focused learning model is then described by:

$$\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \quad \boldsymbol{y} = \tilde{f}\big(r(\boldsymbol{\theta})\big) \tag{11}$$

Note that cost regularization is not an alternative to controlling the perturbation scale: it is a way of controlling it, acting on the cost estimate rather than on the perturbation. Two consequences follow. First, the approach is method-agnostic, in that the same map r can be placed in front of the various studied DFL methods without modifying their gradient estimators or requiring access to their perturbation distribution. Second, and more importantly, the correction called for is per-sample and time-varying, since $\|h_{\boldsymbol{v}}(\boldsymbol{x})\|$ varies both across inputs and over the course of training, whereas ε, β and α are scalar hyperparameters and can only apply a global, static correction. Cost regularization supplies precisely such a correction, at negligible computational cost.

We first consider a regularization r^n already introduced by Sahoo et al. (2022) to address solution invariance in decision-focused learning

Definition 5.1. Let $r^n : \Theta \to \Theta$ be a L^2 normalization of the cost vector estimate, such that:

$$r^n(\boldsymbol{\theta}) = \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \quad \forall \boldsymbol{\theta} \in \Theta \tag{12}$$

Regularization r^n projects the cost vector estimates onto the unit sphere, thereby upper-bounding the value of their stability radius. According to the solution invariance property 2, one has $f(r^n(\boldsymbol{\theta})) = f(\boldsymbol{\theta})$. Moreover by invariance, it is equivalent to scaling the perturbations by $\|\boldsymbol{\theta}\|$.

Since r^n maps every $\boldsymbol{\theta} \in \Theta$ to a vector of unit norm, it cuts both ways: it inflates the norm of a cost vector whose magnitude is already commensurate with the perturbation just as readily as it shrinks a cost vector of larger norm, whose stability radius would otherwise absorb the effect of the perturbation. This asymmetry explains why r^n can underperform: it offers no guarantee against reintroducing the very failure mode the regularization is meant to prevent. We therefore propose a second regularization r^s which has a low impact on cost vectors with norm below a scalar κ, while reducing the norm of those exceeding it to a value below κ.

Definition 5.2. Let $r^s : \Theta \to \Theta$ be a regularization function smoothly rescaling cost vector estimates so that their L^2 norm does not exceed a given positive value κ:

$$r^s(\boldsymbol{\theta}) = \frac{1}{1 + \kappa^{-1} \|\boldsymbol{\theta}\|} \cdot \boldsymbol{\theta} \tag{13}$$

This regularization corresponds to a shrinkage into an L^2 ball of radius κ ensuring the intended balance between rescaling and flexibility outlined above. Moreover by invariance, one has $f(r^s(\boldsymbol{\theta})) = f(\boldsymbol{\theta})$. Here the choice of value for parameter κ balances flexibility and regularization: larger values yield looser bounds on the stability radius.

6 Numerical Experiments

We conduct experiments to validate the effectiveness of the proposed regularization approaches. To this end, the performance of three state-of-the-art DFL methods is evaluated in terms of regret expressed as a percentage, both with and without regularization. The regularized DFL models are described by equation 11, using regularization r^n or r^s. Namely, we study the SPO, DPO, and DBB approaches, relying on a loss function corresponding to the regret L^r defined in equation 4 for the DPO and DBB methods specifically. The DBB approach is preferred over IMLE because it relies on the same gradient approximation equation 9 while being simpler to implement, as its perturbation distribution p is a point mass distribution concentrated on $\boldsymbol{y}$.

We replicate the experimental setup proposed by the state-of-the-art review of Mandi et al. (2024). Models are trained over multiple initialization seeds. Their hyperparameters are optimized according to a separate evaluation dataset by grid search, and their performance is established on a separate test dataset. Details are provided in appendices A.3. The code is available for reproducibility purposes (Author, 2026).

6.1 Solution Stability on Extreme Cases

We introduce a simple optimization problem to illustrate the impact of poor solution stability management on decision-focused learning. This problem is presented in detail in appendices A.4. In that problem, a decision must be made between three vertices A, B and C constituting a triangle polytope on a 2D plane. The true cost vector leads to a ground truth optimal decision A, while decision B generates slight regret and decision C is highly detrimental. We consider three problem categories, with decision C being gradually more detrimental in categories 1, 2 and 3.

Learning appropriate cost vectors for this problem is prone to degeneration under the DFL methods considered. Indeed as presented in Section 4.2, the gradient computation for these methods relies on decision differences to capture how variations in the cost vector shift the solution from sub-optimal choices toward cost-minimizing ones. This approach becomes problematic in the present setting, because decisions A and B lie directly adjacent to the significantly worse decision C. The large cost discrepancy between these options introduces instability in the differentiation process, as observed in

Table 1: Regret as percentage (%) for the three-vertex problem.

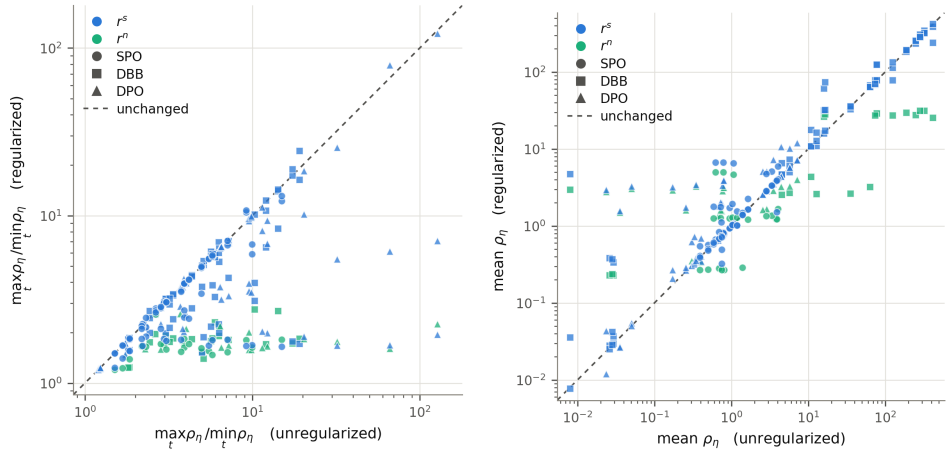
	SPO	SPO-r^n	SPO-r^s	DPO	DPO-r^n	DPO-r^s	DBB	DBB-r^n	DBB-r^s
C1	7.12	*14.04*	**5.51**	34.70	23.07	**21.47**	9.70	*25.57*	**5.82**
C2	10.90	*11.60*	**4.64**	79.32	**16.49**	24.19	23.53	*44.42*	**19.28**
C3	58.70	33.88	**25.81**	266.45	92.89	**72.68**	28.70	*239.54*	**22.57**

our numerical results. Table 1 presents the best performance, in terms of regret, of each DFL method under each regularization method.

Across all standard DFL methods, regret increases with C being gradually more detrimental. Meanwhile, regret remains in all cases substantially lower for DFL methods with shrinkage regularization r^s than for their standard counterpart. First, this corroborates our previous statement which ties the extremely detrimental cost of vertex C with scaling mismatch. Moreover, it demonstrates the relevance of cost regularization to mitigate such issues ; statistical testing supports that claim as shown in Table 4.

We also observe that the cost regularization r^n by normalization performs significantly worse than its shrinkage counterpart r^s and even the standard DFL methods, except in a few cases. As discussed in Section 5, this can be explained by the lack of flexibility induced by this approach in model training. In contrast, the shrinkage regularization r^s performs much better as it allows more flexibility.

We confirm the impact of regularization on DFL training behavior by study of the cost vector stability ratio ρ_η as defined in 4.2, with consideration of $\eta(r(\boldsymbol{\theta}))$ rather than $\eta(\boldsymbol{\theta})$ when regularization applies. Its exact computation is only possible because the optimization problem is particularly simple, and it is detailed in appendices A.4. Figure 1a displays the ratio between the maximum and minimum value of the epoch-averaged ρ_η throughout training, while Figure 1b displays the mean value of ρ_η for the second half of the training epochs.



(a) Max-min ρ_η ratio.

(b) Average ρ_η value.

Figure 1a shows that with either regularization, the max-min ratio varies much less during training. Regularization pulls the ratio closer to 1, and the effect is strongest where the max-min ratio is highest under standard training. We deduce that the cost vector stability radius tends to vary less in regularized setting, which may improve performance. r^s is also more flexible than r^n: the ratio it reaches differs from case to case instead of settling at the same level each time.

Figure 1b shows why this flexibility is key to improving training. When ρ_η is below one under standard training, both regularizations increase it. When ρ_η is above one, r^s leaves it mostly unchanged, whereas r^n decreases it. This decrease comes from the normalization of cost vectors whose norm is below one, and it hurts learning: a lower ratio raises the risk that the optimization mapping is no longer differentiated. Overall, r^s improves training through its flexibility: it raises ρ_η where it is below one and leaves it unchanged otherwise.

Note that for this particularly simple optimization problem, the cost vectors to be learned remain the same for all problem categories, whether or not they are more instability-inducing. A simple way to address instability in this case would be to consider a small b value. However for more realistic optimization problems and to the best of our knowledge, solution instability is much harder to define and mitigate, hence the need for a general-purpose approach such as our proposed cost regularization.

6.2 Solution Stability on Standard Benchmarks

The three-vertex problem was designed to exhibit the detrimental effect of solution stability. We now ask whether such issues arise on standard DFL benchmarks, and what regularization costs when they do not. We consider the shortest-path (SP) and set-matching (SM) problems introduced by Mandi et al. (2024), with their publicly available datasets. For each problem we study three instance categories (SP1–SP3, SM1–SM3), which differ in degree of the feature-to-cost mapping and noise level. Unlike the three-vertex problem, the instability properties of these problems are not known in advance. Table 2 reports the best test regret of each DFL method with and without regularization.

Table 2: Regret as percentage (%) for the shortest-path and set-matching problem.

	SPO	SPO-r^n	SPO-r^s	DPO	DPO-r^n	DPO-r^s	DBB	DBB-r^n	DBB-r^s
SP1	21.24	**15.40**	**15.40**	17.73	**16.08**	16.30	16.55	*17.92*	**16.41**
SP2	**10.07**	*10.12*	*10.40*	11.83	**10.55**	11.18	11.92	11.32	**10.83**
SP3	8.43	**7.67**	*8.50*	125.57	**13.10**	15.72	10.15	**9.63**	*11.29*
SM1	91.72	90.38	**90.09**	92.08	*92.27*	**92.03**	91.53	91.16	**90.62**
SM2	91.90	**89.31**	89.66	92.35	*92.66*	**92.33**	91.19	90.90	**90.17**
SM3	**88.02**	*88.79*	*88.14*	92.35	92.33	**92.05**	90.19	*91.13*	**88.97**

Unlike in the three-vertex problem, the benefit of regularization is not uniform across these problems. It is substantial in some settings, most notably DPO on SP3, where regret drops from 125.57% to 13.10%: a failure of the unregularized method on a standard benchmark, which the shrinkage regularization r^s corrects. In other settings regularization is neutral or slightly detrimental. This mostly concerns the normalization r^n, whose lack of flexibility was already discussed and which we do not recommend in practice. For r^s, the unregularized model performs significantly better in only two cases, SPO on SP2 and DBB on SP3, and by at most 1.14 regret points.

A plausible explanation is that scale mismatch arise only for some combinations of problem and method: where they are absent, regularization has little to correct and can only constrain the model. We stress that this is a hypothesis, not an established fact. Unlike the three-vertex problem, the shortest-path and set-matching problems admit no closed-form expression for the stability radius. We therefore cannot directly check that the settings where regularization helps are those that suffer from instability, and inferring instability from the performance gain alone would be circular.

In summary, on standard benchmarks the shrinkage regularization r^s prevents a severe training failure when one occurs, at a small cost otherwise. Since whether a given problem induces instability cannot be determined beforehand, this makes r^s a sensible default. A broader search over κ may further reduce the residual cost in stable settings.

7 Conclusion

We showed that the scale of cost estimates drifts relative to the perturbations in perturbation-based DFL, which can stall training or reduce it to imitation. Shrinkage regularization of the cost estimates prevents these failures, at little cost when they do not occur.

Our work paves the way for multiple directions of future research. Alternative perturbations such as multiplicative ones (Dalle et al., 2022) could be investigated since they may influence the differentiation process in other ways. Moreover, the generalization bounds recently established by Aubin-Frankowski et al. (2026) for perturbation-smoothed linear oracles offer a promising theoretical avenue for formally grounding the benefits of the regularization approach we demonstrate empirically.

AI USE STATEMENT

In this work, we used generative AI tools to aid writing for literature review. We have not used generative AI tools for research ideation, to draft sections of the paper, for generating synthetic datasets and for proving mathematical claims. We have reviewed all AI-assisted work regarding writing polishing. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All experiments are reproducible within an open-access repository with available datasets.

REFERENCES

Pierre-Cyril Aubin-Frankowski, Yohann De Castro, Axel Parmentier, and Alessandro Rudi. Generalization bounds of surrogate policies for combinatorial optimization problems, 2026. URL https://arxiv.org/abs/2407.17200.

Anonymous Author. Decision-focused learning regularization benchmark. https://anonymous.4open.science/r/DFL_regularization_benchmark-E7BC, 2026.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Mathieu Blondel, André F. T. Martins, and Vlad Niculae. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(1):1–69, 2020.

Joseph F. Bonnans and Alexander Shapiro. *Perturbation Analysis of Optimization Problems*. Springer New York, 2000.

Guillaume Dalle, Léo Baty, Louis Bouvier, and Axel Parmentier. Learning with combinatorial optimization layers: a probabilistic approach, 2022. URL https://arxiv.org/abs/2207.13513.

Justin Domke. Implicit differentiation by perturbation. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2010.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1): 9–26, 2022.

James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2020.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit mle: backpropagating through discrete exponential family distributions. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

George Papandreou and Alan L. Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, 2011.

Axel Parmentier. Learning structured approximations of combinatorial optimization problems. *Open Journal of Mathematical Optimization*, 6:1–27, 2025.

Marin V. Pogancic, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2021.

Subham S. Sahoo, Marin Vlastelica, Anselm Paulus, Vít Musil, Volodymyr Kuleshov, and Georg Martius. Backpropagation through combinatorial algorithms: Identity with projection works. In *International Conference on Learning Representations*, 2022.

Noah Schutte, Senne Berden, Tias Guns, Krzysztof Postek, and Neil Yorke-Smith. Scalable decision-focused learning through cost-sensitive regression, 2026. URL https://arxiv.org/abs/2605.18005.

Yuri N. Sotskov, Vladimir K. Leontev, and Eduard N. Gordeev. Some concepts of stability analysis in combinatorial optimization. *Discrete Applied Mathematics*, 58(2):169–190, 1995.

Beichen Wan and Mo Liu. A solver-free training method for predict-then-optimize, 2026. URL https://arxiv.org/abs/2606.19587.

A Appendix

A.1 Example to Illustrate Solution Stability

Consider the following problem in two dimensions, with linear costs parameterized by a vector $\boldsymbol{\theta} \in \mathbb{R}^2$, and denoted $P(\boldsymbol{\theta})$:

$$P(\boldsymbol{\theta}) : \min \ \boldsymbol{\theta}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{\theta}_2 \cdot \boldsymbol{x}_2 \tag{14}$$

$$\boldsymbol{x}_1 + \boldsymbol{x}_2 = 1 \tag{15}$$

$$0 \leq \boldsymbol{x}_i \leq 1, \qquad \forall i \in \{1, 2\} \tag{16}$$

In this simple problem, the optimal solution depends entirely on the order of costs. Indeed, the optimal solution is $\boldsymbol{x} = (1, 0)$ (resp. $\boldsymbol{x} = (0, 1)$) for all cost vectors such that $\boldsymbol{\theta}_1 < \boldsymbol{\theta}_2$ (resp. $\boldsymbol{\theta}_1 > \boldsymbol{\theta}_2$). Otherwise for all cost vectors such that $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$, any solution $\boldsymbol{x} \in \{(a, 1-a);\ a \in [0, 1]\}$ is optimal.

Consider the set valued optimization mapping $F(\cdot)$ associated to problem P. The set $F(\boldsymbol{\theta})$ is a singleton $\{(1, 0)\}$ (resp. $\{(0, 1)\}$) for all cost vectors such that $\boldsymbol{\theta}_1 < \boldsymbol{\theta}_2$ (resp. $\boldsymbol{\theta}_1 > \boldsymbol{\theta}_2$), and is equal to $\{(a, 1-a);\ a \in [0, 1]\}$ for all cost vectors such that $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$. Hence $F(\boldsymbol{\theta})$ is a singleton for all cost vectors $\boldsymbol{\theta} \in \mathbb{R}^2$ except for those with a direction component along vector $(1, 1)$. We conclude that $F(\cdot)$ is a singleton almost everywhere.

Consider now the optimization mapping $f : \mathbb{R}^2 \to [0, 1]^2$ associated to problem P, such that $f(\boldsymbol{\theta})$ is an optimal solution to problem $P(\boldsymbol{\theta})$ for a cost vector $\boldsymbol{\theta} \in \mathbb{R}^2$. One can observe the invariance property for this particular example. For all cost vectors $\boldsymbol{\theta} \in \mathbb{R}^2$ and positive scalars $\alpha > 0$, one has $f(\alpha \cdot \boldsymbol{\theta}) = f(\boldsymbol{\theta})$ since the order of costs remains the same for vectors $\boldsymbol{\theta}$ and $\alpha \cdot \boldsymbol{\theta}$.

We now determine an explicit formulation to compute the stability radius of any cost vector for that particular problem. To that end, consider a cost vector $\boldsymbol{\theta} \in \mathbb{R}^2$ such that $\boldsymbol{\theta}_1 > \boldsymbol{\theta}_2$ and a perturbed cost vector $\boldsymbol{\theta} + \boldsymbol{\delta} \in \mathbb{R}^2$. The decisions are the same for both cost vectors if and only if the first element of $\boldsymbol{\theta} + \boldsymbol{\delta}$ cannot get smaller than the second element of that perturbed cost vector, in which case the cost order is preserved. The decision flips exactly when $\boldsymbol{\theta}$ crosses the hyperplane $\{\boldsymbol{\theta} \in \mathbb{R}^2;\ \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2\}$. The distance of $\boldsymbol{\theta}$ to that hyperplane is equal to $\frac{|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2|}{\sqrt{2}}$: this is therefore the value of the stability radius $\eta(\boldsymbol{\theta})$.

A.2 Proof of Proposition 4.1

We provide a proof for Proposition 4.1.

We consider a cost vector $\boldsymbol{\theta} \in \Theta$ for which $F(\boldsymbol{\theta})$ is the singleton $f(\boldsymbol{\theta})$: hence we do not place ourselves in the extreme case of cone boundaries where one would have $\eta(\boldsymbol{\theta}) = 0$. Such cases are seldom met in practice during the training phase as it would require to estimate the cost vector precisely on that cone boundary.

Let us first prove the proportional relation. Consider a cost vector $\boldsymbol{\theta} \in \Theta$ and write $\hat{\boldsymbol{\theta}} = \|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}$. Since Θ is a cone, $\mathcal{P}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\| \cdot \mathcal{P}(\hat{\boldsymbol{\theta}})$, and by the invariance property 2, $f(\hat{\boldsymbol{\theta}}) = f(\boldsymbol{\theta})$.

For any perturbation $\boldsymbol{\delta} \in \mathcal{P}(\hat{\boldsymbol{\theta}})$, invariance gives $f(\hat{\boldsymbol{\theta}} + \boldsymbol{\delta}) = f(\boldsymbol{\theta} + \|\boldsymbol{\theta}\| \cdot \boldsymbol{\delta})$. Hence if $\|\boldsymbol{\delta}\| < \eta(\hat{\boldsymbol{\theta}})$, then $f(\boldsymbol{\theta} + \|\boldsymbol{\theta}\| \cdot \boldsymbol{\delta}) = f(\hat{\boldsymbol{\theta}}) = f(\boldsymbol{\theta})$. Every perturbation of $\boldsymbol{\theta}$ with norm below $\|\boldsymbol{\theta}\| \cdot \eta(\hat{\boldsymbol{\theta}})$ therefore leaves the decision unchanged, and $\eta(\boldsymbol{\theta}) \geq \|\boldsymbol{\theta}\| \cdot \eta(\hat{\boldsymbol{\theta}})$.

Conversely, for any perturbation $\boldsymbol{\delta} \in \mathcal{P}(\boldsymbol{\theta})$, invariance gives $f(\boldsymbol{\theta} + \boldsymbol{\delta}) = f(\hat{\boldsymbol{\theta}} + \|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\delta})$. Hence if $\|\boldsymbol{\delta}\| < \eta(\boldsymbol{\theta})$, then $f(\hat{\boldsymbol{\theta}} + \|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\delta}) = f(\boldsymbol{\theta}) = f(\hat{\boldsymbol{\theta}})$. Every perturbation of $\hat{\boldsymbol{\theta}}$ with norm below $\|\boldsymbol{\theta}\|^{-1} \cdot \eta(\boldsymbol{\theta})$ therefore leaves the decision unchanged, and $\eta(\hat{\boldsymbol{\theta}}) \geq \|\boldsymbol{\theta}\|^{-1} \cdot \eta(\boldsymbol{\theta})$.

Combining both inequalities, we conclude that $\eta(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\| \cdot \eta(\hat{\boldsymbol{\theta}})$.

We now prove that $\eta(\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}) \in (0, 1]$. It is assumed that $F(\boldsymbol{\theta})$ is the singleton $f(\boldsymbol{\theta})$: by invariance one has $F(\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}) = F(\boldsymbol{\theta})$ which also is the singleton $f(\boldsymbol{\theta})$, and it follows that $\eta(\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}) > 0$. Moreover, one always has $F(\mathbf{0}) = \mathcal{C}$, hence we deduce that $\eta(\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta}) \leq \|\|\boldsymbol{\theta}\|^{-1} \cdot \boldsymbol{\theta} - 0\| = 1$. This concludes our proof.

A.3 Experimentation Setup and Statistical Testing

All experiments and results can be reproduced with the code provided in the indicated Git repository (Author, 2026).

For each instance of each problem, we consider a learning model described by equation 2 such that $h_{\boldsymbol{v}}(\cdot)$ is a fully connected neural network. For each experiment, ten trials are conducted during both validation and testing phases, with network weights initialized using distinct seeds ranging from 0 to 9. Performance is assessed based on the average regret over the test dataset across these ten training runs. The number of training epochs is fixed at 30, and a grid search is performed to identify the optimal hyper-parameter set for each DFL technique using a designated evaluation dataset.

We propose in Table 3 the value range of each parameter involved in one of the studied learning approaches in Section 6.

Table 3: Value table for the hyper-parameter tuning of all learning approaches

Hyper-parameter	DFL technique	Range
learning rate	All	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$
κ	All	$\{1, 10^2, 10^4\}$
α	SPO	$\{0.5, 2, 10\}$
β	DBB	$\{0.1, 1, 10\}$
ϵ	DPO	$\{0.1, 0.5, 1\}$

All parameter ranges have been selected according to pre-tests leading to the dismissal of exceedingly large or small values, based on numerical experiments proposed by Mandi et al. (2024) that designed this benchmark. Regarding the range of values for the parameter κ in regularization r^s, they act as a measure of the intensity of the regularization, with each value corresponding respectively to a strict, moderate or low regularization. The number of possible values for each parameter range is considered reasonable given the extensive computation time required to train each decision-focused learning model.

For each optimization problem and its associated datasets, the performance of all DFL techniques is reported using their respective optimal hyperparameter configurations, evaluated according to a regret-based metric. For each individual model, the reported performance corresponds to the average regret obtained over ten training runs with random seeds ranging from 0 to 9. Results are summarized in Tables 1 and 2.

We provide information relative to the statistical relevance of our claim stating that the regularized DFL techniques perform better than their standard counterpart.

To assess whether the proposed regularization consistently outperforms the baseline under identical training conditions, we conduct a paired non-parametric analysis across matched random seeds, using

Table 4: Statistical testing for the three-vertex problem study.

	SPO	DPO	DBB
Cat. 1	0.001	0.001	0.001
Cat. 2	0.001	0.001	0.001
Cat. 3	0.001	0.001	0.001

Table 5: Statistical testing for the general study.

	SPO	DPO	DBB
SP1	0.05	0.001	0.001
SP2	*0.001*	0.001	0.001
SP3	*0.07*	0.001	*0.002*
SM1	0.03	0.15	0.05
SM2	0.05	0.32	0.03
SM3	*0.46*	0.53	0.09

the best hyperparameter configurations identified for each model and evaluating performance via the regret computed on the test dataset. We apply a Wilcoxon signed-rank test with a one-sided alternative hypothesis and excluding ties to determine whether the best-performing method consistently surpasses its counterpart, for each problem, dataset and DFL technique. The corresponding p-values are reported in Tables 5 and 4 for the general and in-depth analyses, respectively.

Values in italics indicate p-values testing whether the standard model performs better, while the remaining values test whether the regularized model r^s is superior. The consistently low p-values observed across experiments support our claim that the regularized DFL techniques outperform their standard counterparts in most cases.

As discussed in Section 6, it is not possible to determine a priori whether a given problem or dataset will exhibit instability during DFL training. We assume that, when instability is present, regularization significantly improves performance. Conversely, in cases where training remains stable, regularization offers minimal advantage. This may explain the varying p-values depending on the considered case, as certain DFL techniques are more susceptible to instability for specific problems and datasets than for others.

A.4 Detailed Three-Vertex Problem

We provide more details on the specially designed toy problem introduced in Section 6.1. We consider a linear programming model illustrated by Figure 2.

Consider three vertices $A = (\cos(\frac{4\pi}{3}), \sin(\frac{4\pi}{3}))$, $B = (\cos(0), \sin(0))$ and C along the half-axis $\{b \cdot (\cos(\frac{2\pi}{3}), \sin(\frac{2\pi}{3})) \in \mathbb{R}^2 |\ b \geq 0\}$. In that problem, one has $\Theta = \mathbb{R}^2$ and the linear objective is optimized for decisions in the convex hull $\mathcal{C}$ of vertex A, B and C. The optimal decision is equal to A, B or C, depending on the cone Θ_1, Θ_2 or Θ_3 to which the directional component of the cost vector belongs.

We examine three categories of problems, corresponding to three possible positions of vertex C along the w half-axis, with b of value 1, 25 or 100. For each category, instances are generated by rotation of the polytope around the origin.

For each problem category, a dataset of 300 instances is generated. It is then divided into a training dataset, an evaluation dataset and a test dataset of 100 instances each. Each instance i is obtained by random generation of a true cost vector $\bar{\boldsymbol{\theta}}^i$ in set $\{(\cos(\omega), \sin(\omega)) \in \mathbb{R}^2 |\ \omega \in [|\frac{3\pi}{2}, \frac{10\pi}{6}|]\}$ and of a random rotation angle $\zeta^i \in [|0, 2\pi|]$. Then, for a rotation matrix M^ζ of angle ζ in two dimensions, the exact optimization problem for each instance can be reformulated as a minimization problem

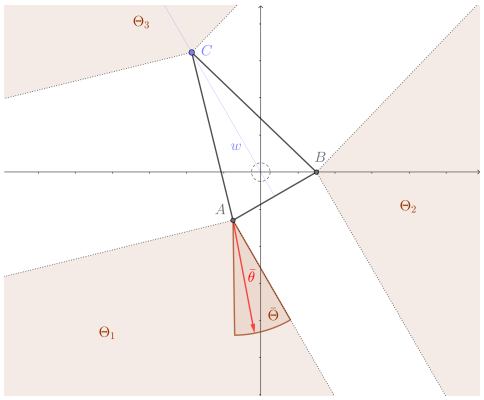


Figure 2: Three-Vertex Problem Illustration

expressed as:

$$\min \quad -(M^{\zeta^i}\bar{\boldsymbol{\theta}}^i)^\intercal \cdot x \tag{17}$$

$$\text{s.t. } x \in \{M^{\zeta^i}A, M^{\zeta^i}B, M^{\zeta^i}C\} \tag{18}$$

For each instance, the true cost vector $\bar{\theta}$ is within set $\bar{\Theta} \subset \Theta_1$, with a corresponding optimal decision A. Likewise for each instance, decision C is greatly detrimental with regard to cost, as it is nearly opposed to the true cost vector $\bar{\theta}$ in set $\bar{\Theta}$. For each category, the ML model is trained on a dataset comprising all related instances where the input consists of rotation angles and the output consists of true cost vectors.

For this particular optimization problem, the stability radius can be expressed in close form. Fix an instance with rotation angle ζ and write $Y^\zeta = \{M^\zeta A,\, M^\zeta B,\, M^\zeta C\}$ for its vertex set, so that $f(\boldsymbol{\theta}) = \arg\max_{x \in Y^\zeta} \boldsymbol{\theta}^\top x$. Let $\boldsymbol{\theta} \in \Theta$ be such that $F(\boldsymbol{\theta})$ is the singleton $\{x^\star\}$, that is, $\boldsymbol{\theta}^\top(x^\star - x) > 0$ for every $x \in Y^\zeta \setminus \{x^\star\}$. For a perturbation $\delta \in P(\boldsymbol{\theta})$ one has $F(\boldsymbol{\theta} + \boldsymbol{\delta}) = F(\boldsymbol{\theta})$ if and only if $(\boldsymbol{\theta} + \boldsymbol{\delta})^\top(x^\star - x) > 0$ for all such x. The perturbations violating this condition for a given x form the closed half-space $\{\boldsymbol{\delta} : \boldsymbol{\delta}^\top(x^\star - x) \le -\boldsymbol{\theta}^\top(x^\star - x)\}$, whose Euclidean distance to the origin is $\boldsymbol{\theta}^\top(x^\star - x)/\|x^\star - x\|$. The stability radius is therefore the smallest of these distances:

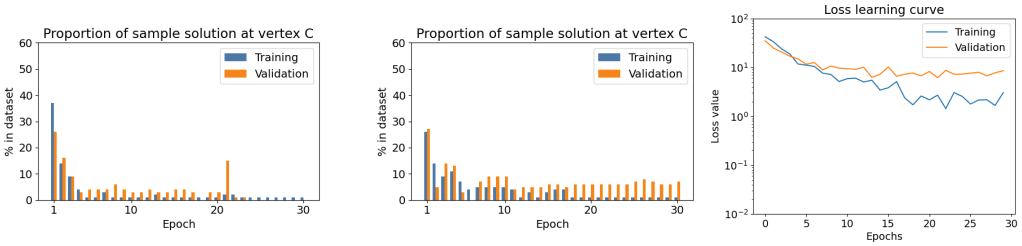
$$\eta(\boldsymbol{\theta}) \;=\; \min_{x \in Y^\zeta \setminus \{x^\star\}} \frac{\boldsymbol{\theta}^\top(x^\star - x)}{\|x^\star - x\|}, \qquad x^\star = f(\boldsymbol{\theta}). \tag{19}$$

We compare the standard and regularized DFL approach on all problem categories, in accordance with the experimental setup detailed in appendices A.3. Results, in terms of regret value, are reported in Table 1: across all problem categories, the regularized formulation consistently achieves superior performance compared with the standard one, with statistical testing supporting that claim as shown in Table 4.

This performance gap can be attributed to scale mismatch inherent to the unregularized model. To substantiate this claim, we provide an in-depth analysis for the DPO approach. Figures 3 displays, for problem category 3 and random seed 6, the proportion of samples whose optimal decision lies at vertex C throughout DPO training, computed on both the training and evaluation datasets and shown for both the standard and regularized variants. The observations for that specific case generalizes to all other observed cases.

For the regularized DPO approach, this proportion in both the training and validation dataset decreases during learning and eventually reaches zero, as expected since assigning a decision at vertex C entails a substantial increase in the loss.

In contrast, under the standard DPO approach, this proportion in the training dataset does not consistently converge to zero, which naturally leads to degraded performance compared to the regularized DPO model. Even in cases where it does vanish in the training set, the corresponding proportion in the validation dataset often remains strictly positive and plateaus at levels substantially higher than those observed during training. This behavior is indicative of overfitting: the model



(a) Regularized DPO model (b) Standard DPO model (c) Standard DPO model

Figure 3: Cat. 3, seed 6, DPO loss: proportion of samples at vertex C and standard model loss values.

fails to reliably learn to map cost vectors outside Θ_3 and instead overfits the few training samples exhibiting this configuration.

We further report the evolution of the DPO loss for the standard model to highlight that this overfitting is atypical. Despite the phenomenon described above, the evaluation loss does not display the usual increase associated with conventional overfitting. Rather than learning from the samples and subsequently overfitting them, the model is unable to learn from certain cases altogether and immediately overfits these specific instances.

We conclude that the unregularized model fails to reliably learn to map cost vectors outside Θ_3 and instead can only overfits the training samples exhibiting this pattern.