Score Function Gradient Estimation to Widen the Applicability of Decision-Focused Learning

MATTIA SILVESTRI*, University of Bologna, Italy
SENNE BERDEN†, KU Leuven, Belgium
GAETANO SIGNORELLI, University of Bologna, Italy
ALI İRFAN MAHMUTOĞULLARI, KU Leuven, Belgium
JAYANTA MANDI, KU Leuven, Belgium
BRANDON AMOS, Meta, USA
TIAS GUNS, KU Leuven, Belgium
MICHELE LOMBARDI, University of Bologna, Italy

Background: Real-world optimization problems often contain parameters that are unknown at solving time. For example, in delivery problems, these parameters may be travel times or customer demands. A common strategy in such scenarios is to first predict the parameter values from contextual features using a machine learning model, and then solve the resulting optimization problem. To train the machine learning model, two paradigms can be distinguished. In *prediction-focused learning*, the model is trained to maximize predictive accuracy. However, this can lead to suboptimal decision-making, because it does not account for how prediction errors affect the quality of the downstream decisions. To address this, *decision-focused learning* (DFL) minimizes a task loss that captures how the predictions affect decision quality.
Objectives: One challenge in DFL is that the task loss has zero-valued gradients when the optimization problem is combinatorial, which hinders gradient-based training. For this reason, state-of-the-art DFL methods use surrogate losses and problem smoothing. However, these methods make specific assumptions about the problem structure (e.g., linear or convex problems with unknown parameters occurring only in the objective function). The goal of our work is to overcome these limitations and extend the applicability of DFL.
Method: We propose an alternative DFL approach that makes only minimal assumptions by combining stochastic smoothing with score function gradient estimation. This makes the approach broadly applicable, including to problems with nonlinear objectives, uncertainty in the constraints, and two-stage stochastic optimization problems.
Results: Our experiments show that our method matches or outperforms specialized methods for the problems they are designed for, while also extending to settings where no existing method is applicable. In addition, our method always outperforms models trained with prediction-focused learning.
Conclusions: In this work we demonstrate that by combining stochastic smoothing and score function gradient estimation to estimate the gradients of a smoothed loss, we can train a machine learning model in a DFL fashion without assuming

*Corresponding Author.
†Corresponding Author.

Authors' Contact Information: Mattia Silvestri, ORCID: 0009-0000-3880-7406, mattia.silvestri94@gmail.com, University of Bologna, Bologna, Italy; Senne Berden, ORCID: 0000-0002-6473-5757, senne.berden@kuleuven.be, KU Leuven, Leuven, Belgium; Gaetano Signorelli, gaetano.signorelli2@unibo.it, ORCID: 0009-0005-5221-0717, University of Bologna, Bologna, Italy; Ali İrfan Mahmutoğulları, ORCID: 0000-0002-8770-8567, irfan.mahmutogullari@kuleuven.be, KU Leuven, Leuven, Belgium; Jayanta Mandi, ORCID: 0000-0003-0629-0099, jayanta.mandi@kuleuven.be, KU Leuven, Leuven, Belgium; Brandon Amos, ORCID: 0000-0003-2884-0119, brandon.amos.cs@gmail.com, Meta, New York, USA; Tias Guns, ORCID: 0000-0002-2156-2155, tias.guns@kuleuven.be, KU Leuven, Leuven, Belgium; Michele Lombardi, ORCID: 0000-0003-4709-8888, michele.lombardi2@unibo.it, University of Bologna, Bologna, Italy.

DOI: 10.1613/jair.1.19498

any structural property of the optimization problem. This approach extends the applicability of DFL to a wider range of optimization problems, including those with uncertainty in the constraints. At the same time, it achieves performance that is competitive with or superior to existing DFL methods when they are applicable.

JAIR Track: Constraint Programming and Machine Learning

JAIR Associate Editor: Ferdinando Fioretto

JAIR Reference Format:
Mattia Silvestri, Senne Berden, Gaetano Signorelli, Ali İrfan Mahmutoğulları, Jayanta Mandi, Brandon Amos, Tias Guns, and Michele Lombardi. 2026. Score Function Gradient Estimation to Widen the Applicability of Decision-Focused Learning. *Journal of Artificial Intelligence Research* 85, Article 10 (February 2026), 34 pages. DOI: 10.1613/jair.1.19498

1 Introduction

Real-world optimization problems often contain parameters that are uncertain at solving time. Consider, for example, a manufacturing company that needs to schedule its production to meet uncertain customer demands, or a delivery company that needs to route its vehicles under uncertain traffic conditions. These kinds of problems can be framed as *predict-then-optimize* problems, consisting of a prediction stage followed by an optimization stage. In the prediction stage, a machine learning (ML) model is used to predict the unknown parameters from correlated contextual features. In the optimization stage, the optimization problem is solved using the predicted parameters. The quality of the resulting decisions is thus highly dependent on the ML model's predictions, making the procedure by which this model is trained particularly important. Two training paradigms can be distinguished: prediction-focused and decision-focused learning.

In *prediction-focused learning* (PFL), the predictive model is trained to maximize the accuracy of the predicted parameters using traditional ML losses like the mean squared error or negative log-likelihood. One downside of this approach is that it can lead to suboptimal decision-making, because it does not account for the ways in which prediction errors can affect the solution to the optimization problem. The accuracy of the predictions does not necessarily align with the quality of the resulting decisions. As an illustrative example, consider a knapsack problem in which the item values are unknown and must be predicted prior to solving. Overestimating the value of high-value items does not alter the optimal solution, since these items would be selected regardless of the overestimation. In contrast, underestimating their value by the same amount could cause the items to be excluded, leading to a different and potentially suboptimal solution.

For this reason, *decision-focused learning* (DFL) instead trains the ML model to maximize the quality of the decisions resulting from the predictions by minimizing a task loss, and thus forms a deeper integration between prediction and optimization. Most DFL approaches are gradient-based, and therefore require backpropagation of the gradient through the optimization problem. While this can be done exactly for convex optimization problems using implicit differentiation (Agrawal, Amos, et al. 2019; Amos and Kolter 2017), combinatorial optimization problems present significant challenges. This is because when the parameters of a combinatorial problem change, the solution either does not change at all, or changes abruptly. Consequently, the partial derivatives of the solution with respect to the parameters are zero almost everywhere, and do not exist at the transition points. This makes the direct application of gradient-based ML techniques unhelpful.

To address this challenge, various techniques have been proposed to obtain non-zero gradients that still reflect decision quality (Berden et al. 2025; Donti et al. 2017; Elmachtoub and Grigas 2022; Elmachtoub, Liang, et al. 2020; Mandi, Bucarey, et al. 2022; Mandi and Guns 2020; Mulamba et al. 2021; Shah et al. 2022a; Tang and Khalil 2024a; Wilder et al. 2019). However, most of these methods place strong assumptions on the structure of the optimization problem, for example, requiring that the predicted parameters appear only in the objective function (and not in the constraints), that the problem's objective function is linear, or that there are no integrality constraints.

In this paper, we present a broadly applicable method that does not make such assumptions. Our method uses stochastic smoothing, by applying random perturbations to the predicted parameters at training time, according to a controllable distribution. This smooths out the loss and gives it informative gradients that can be used to train the predictive model in a decision-focused manner. Moreover, by adjusting the distribution's parameters, the smoothed loss can asymptotically approach the true task loss, so that the location of any minimum can in principle be preserved. Still, computing the gradients of the smoothed loss is not trivial without placing strong restrictions on the downstream problem. To overcome this, we propose to use score function gradient estimation (SFGE) (Williams 1992). This approach only requires the task loss to be bounded, allowing us to compute the gradient of the loss with respect to the parameters, regardless of whether they appear in the objective function, the constraints, or both. Furthermore, this approach can be used regardless of whether the problem is linear or not, and whether the problem contains integrality constraints, or other more involved constraints.

In our experimental evaluation, we start by showing that when the predicted parameters appear solely in the objective function, SFGE is marginally bested by the state of the art, while it still significantly outperforms prediction-focused methods. On the other hand, when the predictions (also) occur in the constraints, SFGE matches or outperforms the state of the art in terms of solution quality. We also demonstrate the effectiveness of our method on two-stage stochastic optimization problems, where it achieves great decision quality with vastly improved efficiency at inference time compared to the use of sample average approximation. Finally, we demonstrate that our method can be effectively applied to a nonlinear combinatorial optimization problem where no existing method can be used, further evidencing its broad applicability.

The remainder of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we formally define the predict-then-optimize problem setting, and the task losses used in the paper. We then introduce our method in Section 4, which we experimentally evaluate in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

In this section we provide an overview of existing DFL methods. We refer the reader to the survey papers by Mandi, Kotary, et al. (2024) and Sadana et al. (2025) for a more comprehensive discussion.

Differentiable Optimization. Many DFL methods make use of differentiable optimization, which is concerned with computing the Jacobian of an optimization problem's optimal solution with respect to the problem's parameters. A pioneering work in differentiable optimization is OptNet by Amos and Kolter (2017), which differentiates through convex quadratic programming problems using implicit differentiation of the Karush-Kuhn-Tucker (KKT) optimality conditions. Later, Agrawal, Barratt, et al. (2019) and Amos (2019, Chapter 7) extended this approach by differentiating the optimality conditions of conic programs. However, these methods are not directly applicable to combinatorial optimization problems, since the solution, and hence the task loss, is piecewise-constant in that setting, making the gradients zero almost everywhere.

Predicting objective parameters. Most existing works study the setting where the ML model only predicts parameters in the objective function of the optimization problem. Wilder et al. (2019) focus specifically on linear programs (LP) and make use of the OptNet method referred to above. However, because the optimal solution of an LP is a piecewise-constant function of its cost coefficients, differentiable optimization cannot be used straightforwardly. To this end, Wilder et al. (2019) analytically smooth the optimization problem by adding the Euclidean norm of the decision variables to the objective function, thereby resolving the zero-valued gradient issue. Similarly, Mandi and Guns (2020) employ log-barrier regularization for smoothing, and differentiate the homogeneous self-dual embedding of the LP, rather than the KKT conditions. For integer linear programs (ILPs) and mixed-integer linear programs (MILPs), both works relax the integrality constraints and thus make use of the LP relaxation of the problem. However, this relaxation can lead to suboptimal solutions in the DFL process, as

we demonstrate Appendix A. To improve on this, Ferber et al. (2020) instead use a cutting planes approach that iteratively tightens the LP relaxation and leads to better decision quality as a result.

Another line of work achieves smoothing implicitly through perturbation, rather than by analytically altering the problem's formulation (Berthet et al. 2020; Niepert et al. 2021; Pogancic et al. 2020). By perturbing the predicted objective parameters, they effectively smooth the mapping from the parameters to the solver's output, leading to a non-zero Jacobian of the expected optimal solution. In (Pogancic et al. 2020), the perturbation is deterministic and creates a linear interpolation of the piecewise-constant loss landscape, which produces informative non-zero gradients. In (Berthet et al. 2020), the perturbations are sampled randomly, and the Jacobian is defined using the expected optimal solution over this noise distribution. These approaches enable the use of differentiable optimization layers, where the optimization problem itself is embedded as a component within a larger neural network architecture. However, unlike the approach we propose, they are not capable of handling problems in which the target parameters appear in the constraints, or nonlinearly in the objective function.

A third line of work studies the use of surrogate losses that still reflect decision quality but offer informative non-zero gradients without requiring the use of differentiable optimization. A seminal paper is that of Elmachtoub and Grigas (2022), which proposed the SPO+ surrogate loss, a convex upper bound of the regret (a measure of decision quality that will be formalized in Section 3) that offers informative subgradients. Other notable examples include losses based on noise-contrastive estimation (Mulamba et al. 2021), learning to rank (Mandi, Bucarey, et al. 2022), and geometric properties of the optimization problem's feasible region (Berden et al. 2025; Tang and Khalil 2024a). Finally, the work by Shah et al. (2022b) proposes to *learn* a surrogate loss from evaluations of the true task loss. To do so, they perform random perturbations of the true parameters in each training example, to fit convex local approximations of the decision loss around that example. These example-specific losses are then used during training. However, since the local losses are trained in isolation and are restricted to a specific functional form, this method may lead to an approximate loss function whose optima do not match those of the true task loss. This is further discussed in Appendix D.

Predicting constraint parameters. To the best of our knowledge, CombOptNet was the first work to enable the integration of an ILP problem with parameterized constraints as a layer of a neural architecture (Paulus et al. 2021). The resulting model can then be trained by minimizing any differentiable loss function. For example, the authors use the mean squared error (MSE) between the predicted and ground-truth solutions. More recently, Hu, J. C. H. Lee, et al. (2023b) introduced a method to train a neural network to predict the cost and constraint parameters by minimizing the post-hoc regret, which is a measure of suboptimality computed after potentially applying a recourse action to ensure feasibility. The proposed approach relies on implicit differentiation and the interior point solver of Mandi and Guns (2020). However, it can be applied only for linear packing and covering problems. In a follow-up work, Hu, J. Lee, et al. (2023) proposed a method to predict parameters in the constraints for two-stage problems tackled via MILP. They consider the LP relaxations of the MILPs in both stages, which, as mentioned earlier, may lead to suboptimal solutions. Hu, J. C. H. Lee, et al. (2023a) later proposed a method applicable to every iteratively solvable problem. Nevertheless, this method assumes a linear predictive model and employs coordinate descent for training, and thus cannot be applied when the predictive model is a more complex neural network. Finally, a recent paper by Mandi, Defresne, et al. (2025) takes a different approach than minimizing the post-hoc regret, by avoiding explicit recourse modeling and instead focusing on reducing infeasibility. They propose two loss functions: one penalizes predicted solutions that are infeasible, while the other penalizes true solutions that become infeasible under the predicted parameters. By tuning the relative weights of these losses, they achieve mostly feasible solutions with small optimality gaps. As this work appeared after our initial submission, we did not compare against it.

3 Problem Setting

We consider a parametric optimization problem:

$$z^\star(y) = \arg\min_{z \in Z(y)} f(z, y) \tag{1}$$

The optimization problem returns an optimal solution $z^\star(y) \in \mathbb{R}^n$, which is a minimizer of the objective function $f(., y)$ within the feasible set $Z(y)$. We make no assumption on $Z(y)$, which can include integrality constraints. The optimization problem can be viewed as a mapping $y \mapsto z^\star(y)$ because $z^\star(y)$ depends on y through $f(., y)$ and $Z(y)$. In predict-then-optimize problems, the true parameter vector y is unknown, but correlated to observable features x. We view both x and y as realizations for two random variables X and Y with a joint distribution $P(X, Y)$. A dataset of samples $\{x_i, y_i\}_{i=1}^N$ drawn from $P(X, Y)$ is assumed to be available, and is used to train an ML model m_ω to predict y based on x. At inference time, the model is invoked to obtain predictions $\hat{y} = m_\omega(x)$, which are then used to obtain a solution vector $z^\star(\hat{y})$.

Equation (1) differs from most predict-then-optimize setups in that no assumption is made on the cost function, and that the parameters are also allowed to appear in the constraints. Note that this also encompasses settings where y appears only in $f(., y)$ or only in $Z(y)$.

Task losses. When the unknown parameters y only occur in the *objective* (i.e., $Z(y)$ is a fixed set Z), the quality of a solution $z^\star(\hat{y})$ can be evaluated in terms of *regret*, also referred to as the smart predict-then-optimize (SPO) loss by Elmachtoub and Grigas (2022), which expresses the suboptimality of $z^\star(\hat{y})$ (the decisions made on the basis of the predicted parameters $\hat{y}$) with respect to the true parameters y:

$$Regret(\hat{y}, y) = f(z^\star(\hat{y}), y) - f(z^\star(y), y) \tag{2}$$

When the unknown parameters occur in the *constraints*, the solution $z^\star(\hat{y})$ may violate the true constraints, i.e., $z^\star(\hat{y}) \notin Z(y)$. For example, production volumes in a manufacturing context might fail to meet demands when those are uncertain. In practical cases, constraint violation must be corrected to recover feasibility, thus changing an infeasible solution $z^\star(\hat{y})$ into a feasible solution $z^\star_{corr}(\hat{y}, y)$. For example, in the manufacturing context mentioned above, this may involve buying additional products from a more expensive source in order to meet demand. In the stochastic optimization literature, this is known as a *recourse action* (Birge and Louveaux 2011). Recourse actions often have an associated cost, which can be formally taken into account by introducing a (problem-specific) penalty function $Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y))$ that expresses the cost of correcting an infeasible solution $z^\star(\hat{y})$ to a feasible solution $z^\star_{corr}(\hat{y}, y)$. With these concepts in place, we are ready to define the *post-hoc regret*, introduced by Hu, J. Lee, et al. (2023) and Hu, J. C. H. Lee, et al. (2023a,b):

$$PRegret(\hat{y}, y) = f(z^\star_{corr}(\hat{y}, y), y) - f(z^\star(y), y) + Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y)) \tag{3}$$

Note that, in what follows, we will not always distinguish between the regret and the post-hoc regret. We will generally only refer to the post-hoc regret, without loss of generality. This can be done because when the solution $z^\star(\hat{y})$ is feasible (which is always the case when $\hat{y}$ occurs only in the objective), the regret and post-hoc regret are identical.

Training objective. The goal is to train ML model m_ω to minimize the *expected* post-hoc regret, leading to the following formulation:

$$\arg\min_{\omega} \mathbb{E}_{x,y \sim P(X,Y)} \left[PRegret(m_\omega(x), y) \right] \tag{4}$$

4 Score Function Gradient Estimation

The challenge in minimizing the expected post-hoc regret (4) through gradient-based training, is that for many problem classes of interest, the gradients of the post-hoc regret are zero-valued. This can be seen by applying the

chain rule:

$$\frac{\partial PRegret(\hat{y}, y)}{\partial \omega} = \frac{\partial PRegret(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \omega} \tag{5}$$

The second factor $\partial\hat{y}/\partial\omega$ relates only to the output of the predictive model and forms no issue. The first factor is more complicated:

$$\frac{\partial PRegret(\hat{y}, y)}{\partial \hat{y}} = \frac{\partial (f(z^\star_{corr}(\hat{y}, y), y) - f(z^\star(y), y) + Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y))}{\partial \hat{y}} \tag{6a}$$

$$= \frac{\partial f(z^\star_{corr}(\hat{y}, y), y)}{\partial \hat{y}} + \frac{\partial Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y))}{\partial \hat{y}} \tag{6b}$$

$$= \frac{\partial f(z^\star_{corr}(\hat{y}, y), y)}{\partial z^\star_{corr}(\hat{y}, y)} \frac{\partial z^\star_{corr}(\hat{y}, y)}{\partial z^\star(\hat{y})} \frac{\partial z^\star(\hat{y})}{\partial \hat{y}} + \frac{\partial Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y))}{\partial z^\star(\hat{y})} \frac{\partial z^\star(\hat{y})}{\partial \hat{y}} + \frac{\partial Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y))}{\partial z^\star_{corr}(\hat{y}, y)} \frac{\partial z^\star_{corr}(\hat{y}, y)}{\partial z^\star(\hat{y})} \frac{\partial z^\star(\hat{y})}{\partial \hat{y}} \tag{6c}$$

The factor $\frac{\partial z^\star(\hat{y})}{\partial \hat{y}}$ occurs in every term of Equation (6c), and measures the change in $z^\star(\hat{y})$ when $\hat{y}$ changes infinitesimally. If the problem is combinatorial, this change is zero almost everywhere since small changes in problem parameters do not affect the optimal solution. This makes the gradient $\frac{\partial PRegret(\hat{y}, y)}{\partial \omega}$ zero almost everywhere, and thus obstructs gradient-based learning.

Stochastic smoothing. To tackle this issue, we apply a stochastic perturbation to the predictor output during training. Formally, at training time the ML model no longer outputs a point estimate $\hat{y}$, but a pair $\theta = (\mu, \sigma)$ defining a Gaussian distribution $p_\theta(\hat{y})$ over the problem parameters, centered on the original point estimate and having a standard deviation equal to σ. This leads to a new, smoothed loss function, as an expectation over $p_\theta(\hat{y})$:

$$L(\theta, y) = \mathbb{E}_{\hat{y} \sim p_\theta(\hat{y})}[\mathcal{L}(\hat{y}, y)] \tag{7}$$

where $\mathcal{L}$ refers to the original task loss, i.e., the (post-hoc) regret.

The smoothness of $L(\theta, y)$ is controlled by σ, as depicted in Figure 1. As σ decreases, the expectation asymptotically converges to the true task loss. As a consequence, the location of every optimum can be preserved, at the cost of less informative gradients. As σ increases, the smoothed loss increasingly diverges from the true task loss. Intermediate values offer a trade-off. It is also possible to treat σ as a learnable parameter, which we do in our experiments, and discuss in more detail in Appendix C. With this setup, the training process is left free to adjust the degree of smoothing so as to minimize the expected loss. In Appendix E, we provide more insight into the learned behavior of σ, and show that it tends to decrease throughout training, which can be interpreted as a shift from exploration to exploitation. Note that even though we use a Gaussian distribution for smoothing, we do not actually assume the data to be normally distributed, nor do we expect to accurately learn the variance in the data: the purpose of $p_\theta(\hat{y})$ is purely that of smoothing the task loss.

Training with gradient estimation. The smoothed loss function $L(\theta, y)$ now has an informative non-zero gradient $\frac{\partial L(\theta, y)}{\partial \theta}$. However, the computation of this gradient is not trivial. To compute an estimate, we utilize score function gradient estimation (SFGE), also known as the REINFORCE algorithm in the context of reinforcement learning

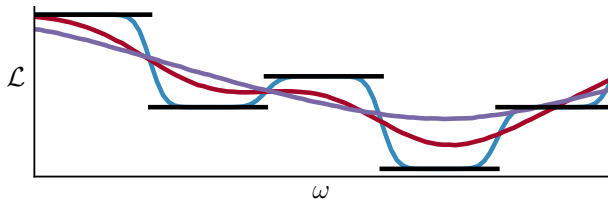

Fig. 1. Illustration of a DFL loss with non-informative zero-valued derivatives (■) smoothed by a Gaussian distribution over the parameters, with increasing variances (■ ≤ ■ ≤ ■). The larger the variance, the more the loss gets smoothed, but the less it resembles the original piecewise-constant task loss.

(Mohamed et al. 2020). Consider the following derivation:

$$\nabla_\theta L(\theta, y) = \nabla_\theta \mathbb{E}_{\hat{y}\sim p_\theta(\hat{y})}[\mathcal{L}(\hat{y}, y)] \tag{8a}$$

$$= \nabla_\theta \int p_\theta(\hat{y})\mathcal{L}(\hat{y}, y)d\hat{y} \tag{8b}$$

$$= \int \mathcal{L}(\hat{y}, y)\nabla_\theta p_\theta(\hat{y})d\hat{y} \tag{8c}$$

$$= \int \mathcal{L}(\hat{y}, y)p_\theta(\hat{y})\nabla_\theta \log p_\theta(\hat{y})d\hat{y} \tag{8d}$$

$$= \mathbb{E}_{\hat{y}\sim p_\theta(\hat{y})}[\mathcal{L}(\hat{y}, y)\nabla_\theta \log p_\theta(\hat{y})] \tag{8e}$$

where (8a) follows from the definition of the loss in (7), (8b) and (8e) follow from the definition of the expectation and (8d) follows the log derivative trick, i.e., the identity $\nabla_\theta \log p_\theta(\hat{y}) = \frac{\nabla_\theta p_\theta(\hat{y})}{p_\theta(\hat{y})}$ whenever $p_\theta(\hat{y}) > 0$. The final gradient in (8e) can be estimated using a Monte Carlo method, giving:

$$\nabla_\theta L(\theta, y) \approx \frac{1}{S}\sum_{i=1}^{S} \mathcal{L}(\hat{y}^{(i)}, y)\nabla_\theta \log p_\theta(\hat{y}^{(i)}) \tag{9}$$

with $\hat{y}^{(i)} \sim p_\theta(\hat{y})$ and S is the total number of samples. We empirically observed that a single sample (i.e., $S = 1$) is typically sufficient, as the gradient noise is mitigated by using stochastic gradient descent. We expand on this in Appendix C. We also note that SFGE is known to suffer from high variance (Greensmith et al. 2004), which can slow down the speed of convergence. To mitigate this issue, we employ *standardization of the regret on each mini-batch*, as also detailed in Appendix C.

The validity of the interchange of the integral and gradient in (8c) requires deeper analysis. As discussed in Mohamed et al. (2020), the interchange is valid if:

(i) $p_\theta(y)$ is continuously differentiable in its parameters θ,
(ii) $p_\theta(\hat{y})\mathcal{L}(\hat{y}, y)$ is both integrable and differentiable for all parameters θ, and
(iii) There exists an integrable function $g(\hat{y})$ such that $\sup_\theta \|\mathcal{L}(\hat{y}, y)\nabla_\theta p_\theta(\hat{y})\|_1 \leq g(\hat{y}), \forall \hat{y}$.

For an arbitrary choice of probability density p_θ and task loss $\mathcal{L}$, it is difficult to check that these three conditions hold (see L'Ecuyer (1995) or Glasserman (1990) for a more detailed discussion). Therefore, we make nonrestrictive assumptions to prove that the above conditions hold for our case.

First, since p_θ is normally distributed, the condition (i) holds due to the smoothness of the density function. For the univariate case:

$$p_\theta(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2}$$

where $\theta = (\mu, \sigma)$. Then, the derivatives of the normal distribution with respect to μ and σ are

$$\frac{y-\mu}{\sigma^2} p_\theta(y) \quad \text{and} \quad \left(\frac{(y-\mu)^2}{\sigma^3} - \frac{1}{\sigma} \right) p_\theta(y), \tag{10}$$

respectively. Both derivatives are continuous in their respective parameters, unless $\sigma = 0$, which can easily be avoided by either using a fixed smoothing parameter σ, or by adding a small positive constant to a trainable, non-negative, σ. The extension to the multivariate case follows directly. Thus (i) holds.

In order to show that (ii) holds, we can also assume that the post-hoc regret $PRegret(\hat{y}, y)$ is bounded. The first part of the post-hoc regret is bounded as it is the difference between the objective values of two feasible solutions if the feasible region of the optimization problem is also bounded. We can also assume that the second part, the penalty, always gives a finite value and admits an upper bound, since in practice there is no infinitely infeasible decision to correct. Then, since $\mathcal{L}$ is bounded, and under the Gaussian assumption, the product p_θ and $\mathcal{L}$ is integrable and differentiable for all parameters θ. Hence (ii) holds under these assumptions.

To show (iii), first note that $\nabla_\theta p_\theta(\hat{y})$ takes a finite value for all θ, which covers our case since μ matches the output of the original point estimator model and σ is controllable. The univariate case is clear in (10) and the extension to the multivariate case is similar. Therefore, $\nabla_\theta p_\theta(\hat{y})$ is bounded, i.e., there exists a (possibly large) positive real number $M(\hat{y})$ depending on $\hat{y}$ such that $\sup_\theta \|\nabla_\theta p_\theta(\hat{y})\|_1 \leq M(\hat{y})$ for all $\hat{y}$.

Using the Cauchy–Schwarz inequality, we get:

$$\|\mathcal{L}(\hat{y}, y) \nabla_\theta p_\theta(\hat{y})\|_1 \leq \mathcal{L}(\hat{y}, y) \|\nabla_\theta p_\theta(\hat{y})\|_1$$

for all θ and $\hat{y}$ since $\mathcal{L}$ is a non-negative real-valued function. Taking the supremum of both sides of the equation with respect to θ, we get:

$$\sup_\theta \|\mathcal{L}(\hat{y}, y) \nabla_\theta p_\theta(\hat{y})\|_1 \leq \mathcal{L}(\hat{y}, y) \sup_\theta \|\nabla_\theta p_\theta(\hat{y})\|_1$$

since the loss does not depend on θ. Then, we have:

$$\sup_\theta \|\mathcal{L}(\hat{y}, y) \nabla_\theta p_\theta(\hat{y})\|_1 \leq g(\hat{y}) := \mathcal{L}(\hat{y}, y) M(\hat{y})$$

where g is constant and hence integrable. Hence (iii) holds.

Our approach differs from existing DFL methods in its broad applicability, due to the fact that Equation (9) makes no assumptions about the optimization problem form or the location of the predicted parameters. In our experimentation, we consider settings *with linear and nonlinear objectives, with and without integrality constraints, and involving uncertain parameters appearing in the objective, in the constraints, or in both.*

Test time. The prediction of a *distribution* $p_\theta(\hat{y})$ through the prediction of μ and σ is introduced solely for the purpose of obtaining informative non-zero gradients. At test time, we directly feed point predictions $\mathbb{E}_{\hat{y} \sim p_\theta}[\hat{y}] = \mu$ from the ML model to the optimization problem. This is in contrast with classical stochastic optimization approaches, which would learn a distribution over parameter vectors, from which multiple realizations would be sampled and incorporated in one large deterministic optimization problem, a process known as sample average approximation (SAA). Our approach thus offers large scalability improvements at inference time with respect to SAA, which we will evidence in our experimental evaluation.

Two notes are in order. The first is that, because of the use of point predictions at test time, our method may not be appropriate for problems whose optimal solutions cannot be identified via a single predicted vector, as studied in Schutte, Veviurko, et al. (2025). In such cases, our formulation will be structurally suboptimal, while scenario-based methods can achieve asymptotic optimality, given unlimited data and computation time. The second is that, this use of point predictions at test time justifies the use of a Gaussian distribution for smoothing, even when the true data distribution is known to be non-Gaussian. After all, only the mean of the predicted

distribution is used at test time. Thus, accurately fitting the remainder of the distribution offers no additional benefits, unless a scenario-based method is employed instead.

5 Experimental Results

To demonstrate the generality of our approach, we conducted the experimental analysis by focusing on four research questions:

(1) How does SFGE compare with PFL and state-of-the-art DFL approaches when predicting parameters appearing exclusively in the constraints, or in both the objective and constraints?
(2) How does SFGE fare against a PFL method that solves a two-stage stochastic optimization approach via SAA at inference time?
(3) How does SFGE compare with PFL and state-of-the-art DFL approaches when predicting parameters exclusively in the objective function?
(4) How does SFGE perform on a problem that falls outside the scope of existing DFL methods?

In our experimental evaluation, we use linear regression models for each method. This is common practice in DFL evaluations and is done to obtain a misspecified predictive model – a setting in which DFL is most promising (Berden et al. 2025; Elmachtoub and Grigas 2022; Mandi, Bucarey, et al. 2022; Mandi, Kotary, et al. 2024; Mandi, Stuckey, et al. 2020; Schutte, Postek, et al. 2023). When utilizing SFGE, the model predicts the mean of a Gaussian distribution, from which we draw one sample of $\hat{y}$ per gradient estimation (i.e., $S = 1$), which we found to work best in practice (see Appendix C). For each dataset, we use a training-validation-test split of 80%, 10%, and 10%, respectively. We refer the reader to the appendices for more details about implementation and related discussions. All the experiments were run on a machine equipped with an Intel(R) Core(TM) i7-1065G7 1.30GHz CPU and 16GB of RAM. All code and data are available at https://github.com/matsilv/sfge-dfl.

5.1 Q1: Predicting Constraint Parameters

We start with the task of predicting parameters in the constraints, a challenging problem that has received limited attention in existing work on DFL.

Linear programs. The approach by Hu, J. C. H. Lee, et al. (2023b) specifically focuses on packing and covering LPs. Thus, in our first experiment, we consider a packing LP – a fractional knapsack problem (KP) with 10 items. In this KP benchmark, both the item values and the item weights are unknown and must be predicted from correlated features. The features are synthetically generated using the procedure described by Hu, J. C. H. Lee, et al. (2023b). This benchmark allows us to compare SFGE against a method that is best-in-class for this specific setting, which we refer to as IntOpt-C in the tables of results.

We use the same recourse action and penalty functions as in Hu, J. C. H. Lee, et al. (2023b): when the solution instantiated by the prediction exceeds the capacity, the amount of each item selected is scaled down until the capacity constraint is satisfied. When the discarded amount of item i is Δ_i, the associated penalty for removing it is $\rho v_i \Delta_i$, where v_i is the item's value. We consider problems with a capacity of 50 and penalty coefficients $\rho = \{0, 1, 2\}$. We conduct experiments on 10 different training-validation-test splits. All methods are trained with Adam (Kingma and Ba 2015), a learning rate of 0.005 and a batch size of 32 samples. Training is stopped when the validation regret (for DFL methods including SFGE) or the validation MSE (for PFL) has stopped improving.

The results are presented in Table 1 (additional results are given in Appendix G). For each method, we report the relative post-hoc regret (*Rel. PRegret*), the relative regret of solutions not requiring recourse actions (*Feas. rel. regret*), the ratio of solutions that require recourse actions (*Infeas. ratio*), the MSE, and the number of epochs until convergence. Both DFL methods outperform PFL in terms of post-hoc regret. Although IntOpt-C has lower post-hoc regret than SFGE, it exhibits notably higher variance. Our method performs slightly worse on average, but with much lower variance. IntOpt-C is fastest in terms of convergence speed, whereas PFL and SFGE are

Table 1. Q1: Predicting Constraints Parameters - Linear programs. PFL, SFGE and PO results on the fractional KP. We omit the *Feas. rel. PRegret* for *Infeas. ratio*s near 1.

Method	*Rel. PRegret*	*Feas. rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
capacity=50, $\rho = 0$					
PFL	0.403 ± 0.015	$\mathbf{0.107 \pm 0.064}$	$\mathbf{0.72 \pm 0.15}$	$\mathbf{99.1 \pm 13.1}$	13.3 ± 2.9
IntOpt-C	$\mathbf{0.377 \pm 0.130}$	–	1.00 ± 0.0	$9.8 \cdot 10^5 \pm 1.2 \cdot 10^4$	$\mathbf{2.5 \pm 1.9}$
SFGE (ours)	0.385 ± 0.008	–	1.00 ± 0.0	$8.2 \cdot 10^5 \pm 6.8 \cdot 10^5$	13.4 ± 3.7
capacity=50, $\rho = 1$					
PFL	0.501 ± 0.033	$\mathbf{0.107 \pm 0.064}$	0.72 ± 0.15	$\mathbf{99.1 \pm 13.1}$	13.3 ± 2.9
IntOpt-C	$\mathbf{0.460 \pm 0.162}$	0.380 ± 0.018	0.61 ± 0.02	$3.8 \cdot 10^5 \pm 4.8 \cdot 10^3$	$\mathbf{2.1 \pm 1.9}$
SFGE (ours)	0.467 ± 0.016	0.177 ± 0.045	$\mathbf{0.55 \pm 0.10}$	$7.9 \cdot 10^5 \pm 5.1 \cdot 10^5$	14.9 ± 4.7
capacity=50, $\rho = 2$					
PFL	0.600 ± 0.077	$\mathbf{0.107 \pm 0.064}$	0.72 ± 0.15	$\mathbf{99.1 \pm 13.1}$	13.3 ± 2.9
IntOpt-C	$\mathbf{0.492 \pm 0.173}$	0.422 ± 0.009	$\mathbf{0.42 \pm 0.05}$	$3.5 \cdot 10^5 \pm 3.8 \cdot 10^3$	$\mathbf{1.6 \pm 0.6}$
SFGE (ours)	0.512 ± 0.036	0.237 ± 0.092	0.46 ± 0.18	$1.3 \cdot 10^6 \pm 9.3 \cdot 10^5$	16.8 ± 4.3

slower and require a comparable number of epochs. As expected, PFL delivers the best MSE, because it is trained for maximal accuracy. With increasing ρ, the DFL methods become more conservative: the infeasibility ratio decreases, but at the cost of a worse relative regret on the feasible solutions.

Integer linear programs. While IntOpt-C is limited to linear packing and covering problems, it has been extended for more general LPs by Hu, J. Lee, et al. (2023). However, many real-world combinatorial optimization problems entail integrality constraints and can be framed as (M)ILPs. Note that SFGE can be applied to (M)ILP problems without modification, because it makes no assumption about the optimization problem's structure. Very few methods are capable of applying the DFL paradigm to predict the constraint parameters of (M)ILP problems. As a baseline representative of this class, we consider CombOptNet (Paulus et al. 2021).

To evaluate SFGE for predicting parameters of constraints in an ILP problem, we considered two setups: the KP with unknown item weights, and the weighted set multi-cover (WSMC) with unknown coverage requirements. The mathematical models for KP and WSMC are provided in Appendix B. To introduce stochasticity, we use the mapping described by Elmachtoub and Grigas (2022) in their shortest path experiment. We set the degree of model misspecification $deg = 5$, the number of input features $p = 5$, and the noise half-width $\bar{\epsilon} = 0.5$. The output of this mapping is then used to parameterize a Poisson distribution, both for the item weights in the KP, and the coverage requirements in the WSMC. For the WSMC, the availability matrices were generated following a set of guidelines by Grossman and Wool (1997) that lead to realistic instances. The set costs are generated uniformly at random from the range $[1, 100]$. We generate five datasets and for each dataset we consider three random splits, and use the same hyperparameter configurations as before. As the relation between features x and problem parameters y is stochastic, for PFL, we switch to predicting a probabilistic model, allowing us to sample multiple scenarios; this model is trained by assuming that the parameters are normally distributed with non-contextual standard deviation, to mimic the fact that knowledge over the ground truth distribution is typically not available in practice. This probabilistic model is trained by minimizing the negative log-likelihood, without considering the task loss.

For the KP, the recourse actions allow for adding new items and discarding previously selected items. Given a penalty coefficient ρ, the value of a newly added item is $\frac{v}{\rho}$, while discarding a previously selected item incurs a

Table 2. Q1: Predicting Constraints Parameters - Integer linear programs. PFL and SFGE results on the KP-50 with uncertain weights. We omit the *Feas. rel. PRegret* for *Infeas. ratio*'s near 1.

Method	*Rel. PRegret*	*Feas. rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
50-items, $\rho = 5$					
PFL	0.168 ± 0.036	$\mathbf{0.001 \pm 0.001}$	0.93 ± 0.03	$\mathbf{7.88 \cdot 10^4 \pm 4.04 \cdot 10^4}$	$\mathbf{34.0 \pm 20.3}$
CombOptNet	0.189 ± 0.068	0.008 ± 0.005	$\mathbf{0.91 \pm 0.02}$	$5.26 \cdot 10^7 \pm 2.26 \cdot 10^7$	41.0 ± 13.4
SFGE(ours)	$\mathbf{0.126 \pm 0.015}$	-	0.98 ± 0.02	$3.62 \cdot 10^5 \pm 5.34 \cdot 10^4$	136.0 ± 10.8
50-items, $\rho = 10$					
PFL	0.319 ± 0.081	$\mathbf{0.001 \pm 0.001}$	0.93 ± 0.03	$\mathbf{7.88 \cdot 10^4 \pm 4.04 \cdot 10^4}$	$\mathbf{34.0 \pm 20.3}$
CombOptNet	0.400 ± 0.146	0.008 ± 0.005	$\mathbf{0.91 \pm 0.02}$	$5.26 \cdot 10^7 \pm 2.26 \cdot 10^7$	41.0 ± 13.4
SFGE(ours)	$\mathbf{0.178 \pm 0.019}$	-	0.99 ± 0.01	$3.71 \cdot 10^5 \pm 6.74 \cdot 10^4$	128.8 ± 19.2
50-items, $\rho = 20$					
PFL	0.615 ± 0.174	$\mathbf{0.001 \pm 0.001}$	0.93 ± 0.03	$\mathbf{7.88 \cdot 10^4 \pm 4.04 \cdot 10^4}$	$\mathbf{34.0 \pm 20.3}$
CombOptNet	0.822 ± 0.302	0.008 ± 0.005	$\mathbf{0.91 \pm 0.02}$	$5.26 \cdot 10^7 \pm 2.26 \cdot 10^7$	41.0 ± 13.4
SFGE(ours)	$\mathbf{0.212 \pm 0.022}$	-	0.99 ± 0.01	$3.73 \cdot 10^5 \pm 6.53 \cdot 10^4$	119.3 ± 26.1

Table 3. Q1: Predicting Constraints Parameters - Integer linear programs. PFL and SFGE results on the WSMC-10×50. We omit the *Feas. rel. PRegret* for *Infeas. ratio*'s near 1.

Method	*Rel. PRegret*	*Feas. rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
10×50, $\rho = 1$					
PFL	2.35 ± 0.85	–	0.98 ± 0.01	$\mathbf{1.78 \cdot 10^5 \pm 3.03 \cdot 10^4}$	$\mathbf{35.3 \pm 44.2}$
CombOptNet	3.04 ± 1.20	–	1.0 ± 0.0	$8.09 \cdot 10^6 \pm 5.24 \cdot 10^6$	49.9 ± 29.5
SFGE (ours)	$\mathbf{1.93 \pm 0.50}$	–	$\mathbf{0.94 \pm 0.05}$	$3.60 \cdot 10^5 \pm 5.88 \cdot 10^4$	70.3 ± 13.1
10×50, $\rho = 5$					
PFL	12.20 ± 4.73	$\mathbf{0.034 \pm 0.019}$	0.96 ± 0.01	$\mathbf{2.01 \cdot 10^5 \pm 3.55 \cdot 10^4}$	61.5 ± 82.4
CombOptNet	88.80 ± 34.3	–	1.0 ± 0.0	$6.34 \cdot 10^6 \pm 2.64 \cdot 10^6$	$\mathbf{45.8 \pm 13.9}$
SFGE (ours)	$\mathbf{4.86 \pm 1.15}$	0.665 ± 0.315	$\mathbf{0.65 \pm 0.08}$	$5.54 \cdot 10^5 \pm 1.25 \cdot 10^5$	81.4 ± 16.9
10×50, $\rho = 10$					
PFL	22.40 ± 7.90	$\mathbf{0.027 \pm 0.041}$	0.98 ± 0.01	$\mathbf{2.18 \cdot 10^5 \pm 7.40 \cdot 10^4}$	72.3 ± 97.5
CombOptNet	374.17 ± 79.0	–	1.0 ± 0.0	$9.10 \cdot 10^6 \pm 3.95 \cdot 10^6$	$\mathbf{34.0 \pm 15.7}$
SFGE (ours)	$\mathbf{7.08 \pm 1.29}$	1.30 ± 0.53	$\mathbf{0.54 \pm 0.14}$	$7.39 \cdot 10^5 \pm 2.21 \cdot 10^5$	67.7 ± 14.4

cost of ρv. For the WSMC, the recourse action consists of adding extra units of the non-covered items at the price of paying an additional cost. The additional cost for each unit of unsatisfied coverage requirement of the i-th item is computed as the maximum set cost among the ones that cover it, multiplied by the coefficient ρ. We run experiments on the KP-50 and with $\rho \in \{5, 10, 20\}$, and on the WSMC with 10 items and 50 sets. In Appendix G, we provide additional results for the WSMC 5×25 and for a KP-50 with stochastic capacity instead of item weights, which lead to similar results.

We can see in Table 2 and Table 3 that *SFGE significantly outperforms the other methods in terms of expected relative post-hoc regret*. In the WSMC, SFGE consistently produces the lowest relative post-hoc regret. As ρ increases, it becomes more conservative, reducing the infeasibility ratio, but resulting in a higher relative regret for solutions that do not require the recourse action. In the case of the KP, the SFGE solutions generally require recourse actions; but still consistently achieve the lowest post-hoc regret. In terms of MSE, PFL is the most accurate, while CombOptNet is the least accurate. Similarly to the previous experiments, SFGE converges slowly, requiring a relatively large number of epochs.

5.2 Q2: Two-Stage Stochastic Optimization

The ILP problems tackled in the previous section involve stochasticity in the ground-truth relation from features to problem parameters. Thus, the performance of the PFL method can be further improved by performing SAA over the predicted distribution *at inference time* (Kleywegt et al. 2002). This involves collecting a set of instance-specific samples, which are used as scenarios in the SAA algorithm to compute the optimal solution $z^\star$. The resulting solution is then employed to calculate the post-hoc regret. We refer to this pipeline as PFL+SAA. In contrast, SFGE relies on stochasticity to smooth the regret, and does not aim to accurately learning the underlying distribution; moreover, our approach relies on *a single sample* during inference. We compare these two methods to explore the scalability advantages of SFGE at inference time, and compare the quality of solutions obtained by both approaches.

In Figures 2 and 3, we present the relative post-hoc regret (top row) and the normalized runtime during inference on a logarithmic scale (bottom row) against the number of sampled scenarios on the same ILP problems as before. For the WSMC we could solve the optimization problem (with scenarios) to optimality; in the case of the KP, we had to impose a time limit of 30 seconds. The corresponding values for SFGE are drawn as horizontal lines since SFGE does not sample scenarios. We can observe that with an increase in the number of scenarios, PFL+SAA generally improves on PFL in terms of relative post-hoc regret, but requires more computation time. We also observe that for high ρ values, *PFL+SAA struggles to catch up to SFGE*. Despite collecting 100 samples for KP and 75 samples for WSMC, PFL+SAA does not outperform SFGE. This is likely due to the model misspecification present in the experiment, as well as the fact that PFL relies on an assumed data distribution that differs from the ground-truth distribution (Gaussian vs. Poisson). Additional results that support our conclusions are provided in Appendix G. These include, for all benchmarks involving a recourse action, an evaluation performed by sampling multiple realizations for every instance in the dataset. The additional results confirm the trends observed above.

5.3 Q3: Predicting Objective Parameters

In the following experiment, we consider the task of predicting parameters that appear linearly in the objective function, a setting that most existing DFL methods have focused on. Since these methods are designed for this task, we do not expect SFGE to surpass them. Our aim is to assess how well SFGE performs compared to them.

We use the 0-1 KP with 50 items with unknown item values. We generate synthetic data by using the same mapping between input features and targets as described in the previous section, along with the same evaluation procedure. We again use the data generation process from Elmachtoub and Grigas (2022) to introduce randomness into how features are mapped to item values. We generate five datasets and for each dataset we consider three random splits. We use the same hyperparameter configurations as before.

In this setup, the SPO+ (Elmachtoub and Grigas 2022) and DPO (the Fenchel-Young alternative) (Berthet et al. 2020) losses will be used as a reference DFL methods for comparison: SPO+ has shown great performance across the board in existing comparative analyses (Mandi, Kotary, et al. 2024; Tang and Khalil 2024b) while DPO relies on smoothing through perturbation, as our method does. When applying DPO we rely on a Gumbel distribution

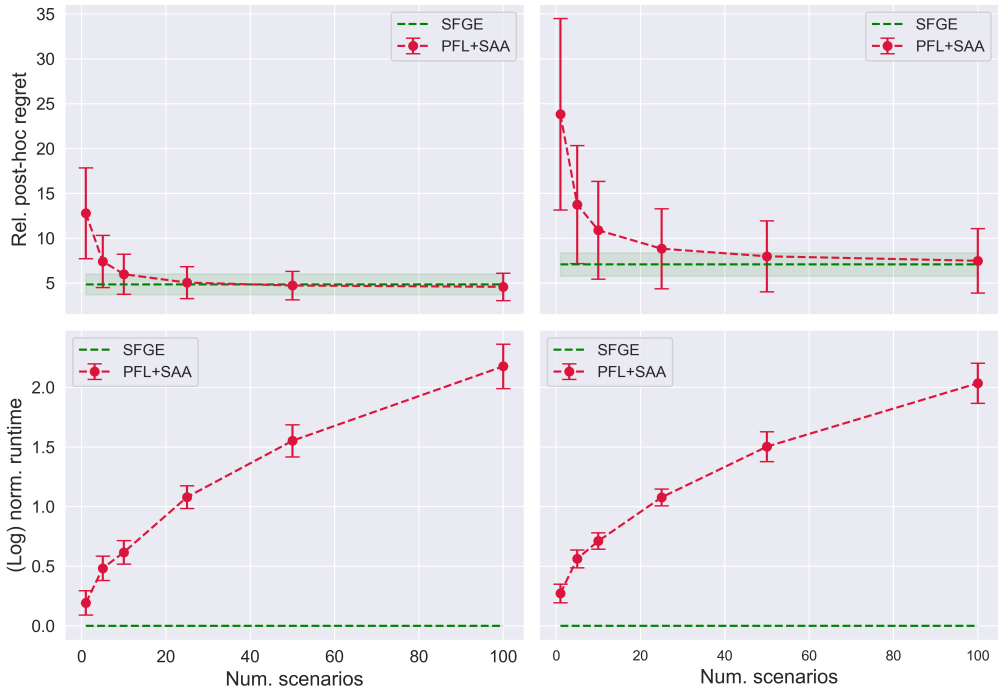

Fig. 2. Q2: Two-Stage Stochastic Optimization. The relative post-hoc regret and normalized runtime at inference time of SFGE and PFL+SAA on the WSMC (size 10×50), for $\rho = 5$ (left) and $\rho = 10$ (right).

Table 4. Q3: Predicting Objective Parameters. PFL, SFGE and SPO results on the KP-50 with uncertain objective coefficients.

Method	*Rel. regret*	*MSE* ($\times 10^4$)	*Epochs*
PFL	0.022 ± 0.005	**2.38 ± 1.28**	**40.5 ± 28.5**
SPO	0.003 ± 0.0004	4.12 ± 1.52	**40.3 ± 9.77**
DPO	**0.002 ± 0.001**	6.24 ± 3.30	45.0 ± 13.4
SFGE	0.009 ± 0.002	17.5 ± 3.34	149.8 ± 15.1
SFGE-MAP	0.008 ± 0.002	10.7 ± 2.03	83.1 ± 8.03

to perturb the solution with a scale factor $\epsilon = 0.1$, and we approximate $dz^\star/dy$ with a single Monte Carlo sample. We also employ a baseline PFL model, trained to minimize the MSE between the predictions and targets.

Convergence speed is important for DFL methods, since for each gradient update step we need to solve an optimization problem, which can be computationally expensive thus hindering its application to real-world scenarios. To improve convergence speed, inspired by Mulamba et al. (2021), we propose an alternative task loss for problems where predictions occur linearly in the objective function. The task loss is:

$$\mathcal{L}(\hat{y}, y) = Regret(\hat{y}, y) + f(\hat{y}, z) - f(\hat{y}, \hat{z}) \tag{11}$$

where f is the objective function. We add the regret with respect to the predicted solution to the original task loss (the $\mathcal{L}_{MAP}$ as described in Mulamba et al. (2021)). We refer to this method as SFGE-MAP.

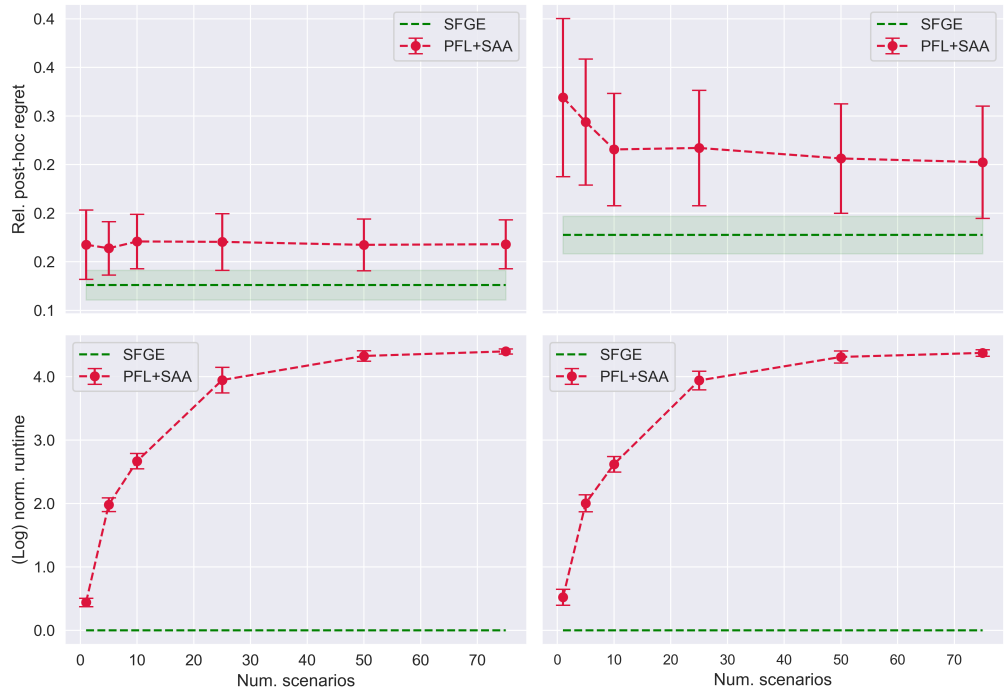

Fig. 3. Q2: Two-Stage Stochastic Optimization. Comparison between SFGE and PFL+SAA on KP-5, for $\rho = 5$ (left) and $\rho = 10$ (right).

The aggregated results are reported in Table 4. Looking at the relative regret, we see that SPO performs best, but that SFGE is still able to outperform PFL significantly. In terms of convergence speed, SPO and PFL require a comparable number of epochs whereas SFGE is significantly slower to converge. We also observe that SFGE-MAP improves convergence speed by a noticeable margin and should therefore, when applicable, be preferred over its standard counterpart.

Moreover, in Table 6 of Appendix G we present additional results for the KP-50 dataset and for quadratic KP instances, with similar outcomes. In conclusion, it appears that SFGE may not be the best choice when the uncertain parameters represent coefficients of a linear cost function; existing DFL methods can effectively address this particular problem in less training epochs. However, the fact that SFGE outperforms PFL also in this case allows us to position SFGE as a generic DFL method, capable of tackling a wide range of predict-then-optimize problems.

5.4 Q4: Widening Applicability

Existing approaches are limited by the specific assumptions they make on the optimization problem's structure. Conversely, SFGE makes no assumptions about problem structure, and can thus straightforwardly be applied to any setting, and can be combined with any (black-box) solver.

To evidence this advantage, we performed an experimental evaluation on a problem for which, to the best of our knowledge, no other DFL approach can be successfully applied, showing how SFGE can be the only candidate solution to surpass PFL in certain use cases. Concretely, we considered a production planning problem

Table 5. Q4: Widening Applicability. PFL and SFGE results on the production planning problem.

Method	*Rel. regret*	*MSE*	*Epochs*
PFL	211.26 ± 33.01	**21.44 ± 0.57**	**142.16 ± 24.12**
SFGE	**116.04 ± 14.35**	39.65 ± 0.15	246.08 ± 12.53

characterized by a quadratic objective function. Given a set of n products $p_1, p_2, \ldots, p_n$, each with an associated overproduction and underproduction cost o_i and u_i, and a maximum capacity c, the goal is to find the optimal integer production values $v \in \mathbb{Z}_+^n$ to minimize the overall cost, computed as:

$$\mathrm{f}(z, y) = \sum_{i=1}^{n} o_i \max(0, z_i - y_i)^2 + u_i \max(0, y_i - z_i)^2,$$

with $y \in \mathbb{Z}_+^n$ being the (predicted) products demands. This combinatorial problem does not offer informative gradients, due to the integrality constraints. In addition, the nonlinearity of the objective function does not allow the use of methods utilizing an LP relaxation (e.g., Hu, J. C. H. Lee, et al. (2023b)).

To generate the dataset, we sample input features $x \in \mathbb{R}^m$ and a matrix of weights $W \in \mathbb{R}^{n \times m}$ from bounded uniform distributions. Then, given k potential customers, we compute ground-truth demands as:

$$y \sim Bin(k, \sigma(Wx)),$$

with σ being the sigmoid function. This process defines a stochastic setting that correlates demands and observable features. As a simple yet demonstrative case, we set $n = 10$, $m = 4$, $k = 100$, $c = 400$. We create an asymmetry between overproduction and underproduction costs by sampling u from $[0.05, 0.3] \cup [0.7, 0.95]$ and setting $o = 1 - u$.

Results are reported in Table 5. SFGE significantly outperforms PFL in terms of regret, as PFL converges towards the mean value of y given x, which does not lead to optimal decision quality. In contrast, SFGE learns to appropriately underestimate or overestimate demands, according to the corresponding underproduction and overproduction costs.

6 Conclusions and Future Directions

This work widens the applicability of DFL by proposing a method that *does not assume structural properties of the task at hand*. Concretely, we employ stochastic smoothing and SFGE to estimate the gradients of a smoothed loss. This allows the method to be applied to linear and nonlinear problems, with or without integrality constraints, and with uncertainty in the objective, in the constraints, or in both. As a specific use case, we show how our method can be used to address two-stage stochastic problems by using only a single predicted parameter vector and a deterministic optimization model, without the need for SAA. Our experimental evaluation reveals that, for problems with uncertainty in the constraints, SFGE matches or outperforms existing methods in terms of post-hoc regret. When predicting parameters that appear linearly in the objective function, SFGE does not outperform the state of the art, but still provides a major improvement over PFL approaches. The main drawback of our method is that it tends to be slower in convergence speed compared to existing methods. This can partly be attributed to the high variance in the Monte Carlo gradient estimates. We alleviate this drawback by standardizing the regret on each mini-batch, as a way to reduce variance.

An interesting direction for future work is to employ SFGE on a real-world case study. The experiments in this paper were conducted on well-defined synthetic benchmarks from the literature to allow for a controlled evaluation. In practice, real datasets often offer weaker signal, more noise, and may lead to higher variance in the SFGE estimator. To account for this, alternative variance reduction techniques from the literature can be used, for

example, employing an actor-critic style algorithm. Another interesting direction is to investigate the merit of other Monte Carlo gradient estimators, such as the measure-valued gradient estimator, as an alternative to SFGE.

Acknowledgments

This research received funding from the European Research Council (ERC) (Grant No. 101002802, CHAT-Opt), from the European Union's HORIZON-CL4–2021-HUMAN-01 research and innovation programme (Project Tuples, Grant Agreement No. 101070149), and from the Research Foundation Flanders (FWO) project G0G3220N. Senne Berden is a fellow of the Research Foundation – Flanders (FWO-Vlaanderen, 11PQ024N).

References

A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. 2019. "Differentiable Convex Optimization Layers." In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., Vancouver, Canada. https://proceedings.neurips.cc/paper_files/paper/2019/file/9ce3c52fc54362e22053399d3181c638-Paper.pdf.

A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. 2019. "Differentiating Through A Cone Program." *J. Appl. Numer. Optim*, 1, 2, 107–115.

B. Amos. 2019. "Differentiable optimization-based modeling for machine learning." Ph.D. Dissertation. Carnegie Mellon University, Pittsburgh, PA 15213.

B. Amos and J. Z. Kolter. June 2017. "Optnet: Differentiable optimization as a layer in neural networks." In: *Proceedings of the 34th International Conference on Machine Learning* (Proceedings of Machine Learning Research). Ed. by D. Precup and Y. W. Teh. Vol. 70. PMLR, Sydney NSW, Australia, (June 2017), 136–145.

S. Berden, A. I. Mahmutogullari, D. Tsouros, and T. Guns. 2025. "Solver-Free Decision-Focused Learning for Linear Optimization Problems." *arXiv preprint arXiv:2505.22224*.

Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. 2020. "Learning with differentiable pertubed optimizers." *Advances in neural information processing systems*, 33, 9508–9519.

J. R. Birge and F. Louveaux. 2011. *Introduction to Stochastic Programming*. (2nd ed.). Springer Publishing Company, Incorporated. ISBN: 1461402360.

P. Donti, B. Amos, and J. Z. Kolter. 2017. "Task-based End-to-end Model Learning in Stochastic Optimization." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3fc2c60b5782f641f76bcefc39fb2392-Paper.pdf.

A. N. Elmachtoub and P. Grigas. 2022. "Smart "Predict, then Optimize"." *Management Science*, 68, 1, 9–26.

A. N. Elmachtoub, J. C. N. Liang, and R. Mcnellis. 13–18 Jul 2020. "Decision Trees for Decision-Making under the Predict-then-Optimize Framework." In: *Proceedings of the 37th International Conference on Machine Learning* (Proceedings of Machine Learning Research). Ed. by H. D. III and A. Singh. Vol. 119. PMLR, (13–18 Jul 2020), 2858–2867.

A. M. Ferber, B. Wilder, B. Dilkina, and M. Tambe. 2020. "MIPaaL: Mixed Integer Program as a Layer." In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 1504–1511.

P. Glasserman. 1990. *Gradient estimation via perturbation analysis*. Vol. 116. Springer Science & Business Media.

E. Greensmith, P. L. Bartlett, and J. Baxter. 2004. "Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning." *J. Mach. Learn. Res.*, 5, 1471–1530.

T. Grossman and A. Wool. 1997. "Computational experience with approximation algorithms for the set covering problem." *European journal of operational research*, 101, 1, 81–92.

X. Hu, J. Lee, and J. Lee. 2023. "Two-Stage Predict+Optimize for MILPs with Unknown Parameters in Constraints." In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., New Orleans, LA 70130, United States, 14247–14272. https://proceedings.neurips.cc/paper_files/paper/2023/file/2e14be0332c04c76742710e417cedb2a-Paper-Conference.pdf.

X. Hu, J. C. H. Lee, and J. H. M. Lee. 2023a. "Branch & Learn with Post-hoc Correction for Predict+Optimize with Unknown Parameters in Constraints." In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Ed. by A. A. Cire. Springer Nature Switzerland, Cham, 264–280. ISBN: 978-3-031-33271-5.

X. Hu, J. C. H. Lee, and J. H. M. Lee. 2023b. "Predict+Optimize for Packing and Covering LPs with Unknown Parameters in Constraints." In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by B. Williams, Y. Chen, and J. Neville. AAAI Press, 3987–3995.

D. P. Kingma and J. Ba. 2015. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun.

A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello. 2002. "The sample average approximation method for stochastic discrete optimization." *SIAM Journal on Optimization*, 12, 2, 479–502.

P. L'Ecuyer. 1995. "Note: On the interchange of derivative and expectation for likelihood ratio derivative estimators." *Management Science*, 41, 4, 738–747.

J. Mandi, V. Bucarey, M. M. K. Tchomba, and T. Guns. 17–23 Jul 2022. "Decision-Focused Learning: Through the Lens of Learning to Rank." In: *Proceedings of the 39th International Conference on Machine Learning* (Proceedings of Machine Learning Research). Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. PMLR, (17–23 Jul 2022), 14935–14947.

J. Mandi, M. Defresne, S. Berden, and T. Guns. 2025. *Feasibility-Aware Decision-Focused Learning for Predicting Parameters in the Constraints*. (2025). https://arxiv.org/abs/2510.04951 arXiv: 2510.04951 (cs.LG).

J. Mandi and T. Guns. 2020. "Interior Point Solving for LP-based prediction+optimisation." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 7272–7282.

J. Mandi, J. Kotary, S. Berden, M. Mulamba, V. Bucarey, T. Guns, and F. Fioretto. 2024. "Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities." *J. Artif. Intell. Res.*, 80, 1623–1701.

J. Mandi, P. J. Stuckey, T. Guns, et al.. 2020. "Smart predict-and-optimize for hard combinatorial optimization problems." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 02. Vol. 34, 1603–1610.

S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. 2020. "Monte Carlo Gradient Estimation in Machine Learning." *J. Mach. Learn. Res.*, 21, 132, 1–62.

M. Mulamba, J. Mandi, M. Diligenti, M. Lombardi, V. Bucarey, and T. Guns. 2021. "Contrastive Losses and Solution Caching for Predict-and-Optimize." In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. Ed. by Z. Zhou. ijcai.org, 2833–2840.

M. Niepert, P. Minervini, and L. Franceschi. 2021. "Implicit MLE: backpropagating through discrete exponential family distributions." *Advances in Neural Information Processing Systems*, 34, 14567–14579.

A. Paulus, M. Rolinek, V. Musil, B. Amos, and G. Martius. 18–24 Jul 2021. "CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints." In: *Proceedings of the 38th International Conference on Machine Learning* (Proceedings of Machine Learning Research). Ed. by M. Meila and T. Zhang. Vol. 139. PMLR, (18–24 Jul 2021), 8443–8453.

M. V. Pogancic, A. Paulus, V. Musil, G. Martius, and M. Rolinek. 2020. "Differentiation of Blackbox Combinatorial Solvers." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

U. Sadana, A. Chenreddy, E. Delage, A. Forel, E. Frejinger, and T. Vidal. 2025. "A survey of contextual optimization methods for decision-making under uncertainty." *European Journal of Operational Research*, 320, 2, 271–289.

N. Schutte, K. Postek, and N. Yorke-Smith. 2023. "Robust losses for decision-focused learning." *arXiv preprint arXiv:2310.04328*.

N. Schutte, G. Veviurko, K. Postek, and N. Yorke-Smith. 2025. "Sufficient Decision Proxies for Decision-Focused Learning." *arXiv preprint arXiv:2505.03953*.

S. Shah, K. Wang, B. Wilder, A. Perrault, and M. Tambe. 2022a. "Decision-Focused Learning without Decision-Making: Learning Locally Optimized Decision Losses." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 1320–1332. https://proceedings.neurips.cc/paper_files/paper/2022/file/0904c7edde20d7134a77fc7f9cd86ea2-Paper-Conference.pdf.

S. Shah, K. Wang, B. Wilder, A. Perrault, and M. Tambe. 2022b. "Decision-focused learning without decision-making: Learning locally optimized decision losses." *Advances in Neural Information Processing Systems*, 35, 1320–1332.

B. Tang and E. B. Khalil. 2024a. "Cave: A cone-aligned approach for fast predict-then-optimize with binary linear programs." In: *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 193–210.

B. Tang and E. B. Khalil. Sept. 2024b. "PyEPO: A PyTorch-Based End-to-End Predict-Then-Optimize Library for Linear and Integer Programming." *Mathematical Programming Computation*, 16, 3, (Sept. 2024), 297–335.

B. Wilder, B. Dilkina, and M. Tambe. 2019. "Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization." In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*. AAAI Press, 1658–1665.

R. J. Williams. 1992. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." *Mach. Learn.*, 8, 229–256. doi:10.1007/BF00992696.

A LP Relaxations May Fail to Guide DFL Methods towards an Optimum

We illustrate that replacing a MILP problem with its LP relaxation can cause a mismatch between the optimal predicted parameters for the true task loss and the corresponding approximation, as mentioned in section 2. In particular, consider the following simple mixed-integer linear program:

$$\min\ y_0 z_0 + y_1 z_1 \qquad (y) \tag{12}$$

$$\text{s.t. } 2(u-\varepsilon)z_0 + z_1 \geq u - \varepsilon \qquad (c_0) \tag{13}$$

$$-2\varepsilon z_0 + z_1 \geq -\varepsilon \qquad (c_1) \tag{14}$$

$$z_0 \geq 0 \qquad (c_2) \tag{15}$$

$$z_0 \leq 1 \qquad (c_3) \tag{16}$$

$$z_1 \leq u \qquad (c_4) \tag{17}$$

$$z_0 \in \mathbb{Z} \tag{18}$$

where the constraints are labeled as c_0, c_1, c_2, c_3 and c_4 and $\varepsilon > 0$ is a small real number. Assume that the ground truth cost coefficients are given as $y = (0, 1)$.

Note that this problem matches a traditional predict-then-optimize setup, with unknown parameters y appearing in a linear cost function. Figure 4a visually depicts the feasible space (in green) and the optimal solution with respect to the ground truth cost coefficients for the MILP. Figure 4b provides the same information (in blue) for the corresponding LP relaxation, obtained by dropping the integrality constraint.

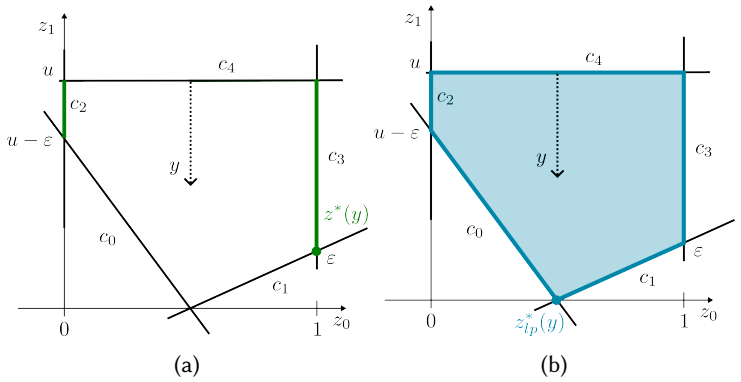

Fig. 4. Feasible space and optimum for the example MILP (a) and its LP relaxation (b)

When the ML model manages to estimate the ground-truth parameter vector y correctly, the regret is 0 for both the MILP and the LP formulations, as expected. However, an incorrect prediction may still lead to the correct optimal solution.

Figure 5a visually depicts the range of predictions that result in the correct LP optimum. If the ML model outputs any of these vectors, the regret (on the LP relaxation) will still be 0. Figure 5b provides a similar analysis for the MILP formulation, i.e., the one representing the true task loss rather than an approximation. Any predicted vector with a direction on the right-hand side of $\tilde{y} = (u - 2\varepsilon, 1)$ – the green-shaded angle – will lead to the correct optimal solution.

There is some overlap between the set of optimal MILP predictions and the set of optimal LP predictions. The larger this overlap, the more effective the use of the LP relaxations in guiding DFL towards high-quality solutions.

However, as depicted in Figure 5c, the set of optimal LP predictions also includes vectors that result in a MILP solution with significantly higher cost. In the figure, the orange angle identifies the parameter vectors that result in an incorrect MILP optimum, which overlaps with the blue angle of optimal predictions for the LP relaxation. The larger this overlap, the less effective LP relaxations become.

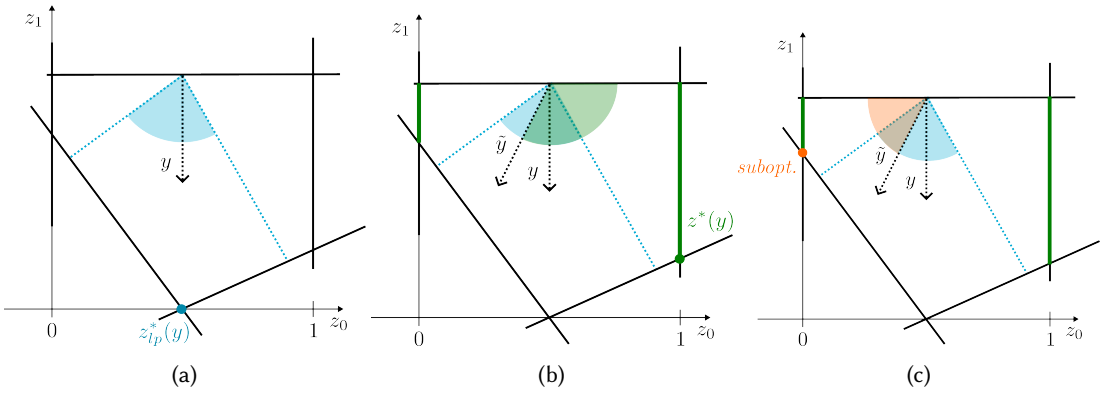

Fig. 5. (a) Range of equivalent optimal predictions for the LP relaxations; (b) range of predictions leading to the correct MILP optimum; (c) range of predictions leading to a suboptimal MILP solution although they are optimal for the LP relaxation.

B Mathematical Models of the Optimization Problems

Mathematical Model of the Knapsack Problem. Given a set of items $\mathcal{I}$, let w_i be the weight of item $i \in \mathcal{I}$. Also, let v_i be the value of item $i \in \mathcal{I}$. The Knapsack Problem (KP) aims to maximize the total value while ensuring that the total weight of selected items does not exceed a given capacity W. The following mathematical model solves the KP:

$$\max_{z} \sum_{i \in \mathcal{I}} v_i z_i \tag{19}$$

$$\text{s.t.} \sum_{i \in \mathcal{I}} w_i z_i \leq W \tag{20}$$

$$z_i \in \{0, 1\} \text{ for all } i \in \mathcal{I}. \tag{21}$$

where the binary variable z_i indicates if item i is selected. The objective (19) maximizes the value of selected items. Constraint (20) ensures that the capacity is respected. In the quadratic KP, the objective is replaced by $\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} v_{ij} z_i z_j$. In the fractional version of the problem, the domain constraint $z_i \in \{0, 1\}$ is replaced by $z_i \in [0, 1]$.

Mathematical Model of the Stochastic Knapsack Problem. The KP with unknown items' weights is a 2-stage stochastic optimization problem. Given the predicted weights $\hat{w}$, we compute the optimal solution $\hat{z}$ by solving the optimization problem described in eqs. (19) to (21). During the second stage, we need to find the optimal

recourse actions that maximize the value of the selected items by solving the following optimization problem:

$$\max_{u^+,u^-} \quad \sum_{i\in\mathcal{I}} \frac{1}{\rho} v_i u_i^+ - \rho v_i u_i^- \tag{22}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}} w_i(\hat{z}_i + u_i^+ - u_i^-) \leq C \tag{23}$$

$$\hat{z} \geq u^- \tag{24}$$

$$\hat{z} + u^+ \leq 1 \tag{25}$$

$$u^+, u^- \in \{0,1\} \tag{26}$$

where u^+ and u^- are respectively the selected/removed items during the second stage, w is the realization of the items' weights and $\rho > 1$ is the penalty coefficient.

The SAA involves solving the first and second stages in a single model. The first stage decisions are the same for all the scenarios, while we need a set of recourse actions for each scenario. The resulting model is:

$$\max_{z,u^+_\omega,u^-_\omega} \quad \sum_{i\in\mathcal{I}} v_i z_i + \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\frac{1}{\rho} v_i u^+_{i,\omega} - \frac{1}{|\Omega|}\sum_{\omega\in\Omega} \rho v_i u^-_{i,\omega} \tag{27}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I},\omega\in\Omega} w_{i,\omega}(z_i + u^+_{i,\omega} - u^-_{i,\omega}) \leq C \qquad \forall\omega\in\Omega \tag{28}$$

$$z \geq u^-_\omega \qquad \forall\omega\in\Omega \tag{29}$$

$$z + u^+_\omega \leq 1 \qquad \forall\omega\in\Omega \tag{30}$$

$$z, u^+_\omega, u^-_\omega \in \{0,1\}, \omega\in\Omega \tag{31}$$

where $\omega \in \Omega$ are the sampled scenarios. When the capacity is uncertain, the model is similar except for the fact that the weights w are known and the capacity is sampled for each scenario.

Mathematical Model of Weighted Set Multi-Cover Problem. Let $\mathcal{I}$ be the set of items and $\mathcal{J}$ be the set of covers. The parameter a_{ij} is 1 if j can cover i and 0 otherwise. Item $i \in \mathcal{I}$ must be covered at least d_i times. The cost of selecting cover $j \in \mathcal{J}$ is c_j. The weighted set multi-cover problem (WSMC) aims to satisfy coverage constraints while minimizing the total cost. The following mathematical model solves the WSMC:

$$\min \sum_{j\in\mathcal{J}} c_j z_j \tag{32}$$

$$\text{s.t.} \sum_{j\in\mathcal{J}} a_{ij} z_j \geq d_i \qquad \forall i\in\mathcal{I} \tag{33}$$

$$z_j \geq 0 \text{ and integer} \qquad \forall j\in\mathcal{J}$$

where the non-negative integer variable z_j indicates how many times cover j is selected. The objective (32) minimizes the total cost. Constraint (33) ensures the coverage requirement for each item.

Mathematical Model of the stochastic WSMC. In our formulation of the stochastic WSMC, the items' coverage requirement is unknown at solution time, resulting in a two-stage stochastic optimization model that can be

formulated as follows:

$$\min \sum_{j \in \mathcal{J}} c_j z_j + \sum_{i \in \mathcal{I}} \rho s_i \tag{34}$$

$$\sum_{j \in \mathcal{J}} a_{i,j} z_j \geq d_i (1 - w_i) \qquad \forall i \in \mathcal{I} \tag{35}$$

$$w_i = 1 \implies s_i \geq d_i - \sum_{j \in \mathcal{J}} a_{i,j} x_j \qquad \forall i \in \mathcal{I}$$

$$z_j \geq 0 \tag{36}$$

$$w_i \in [0, 1] \tag{37}$$

$$s_i \geq 0 \tag{38}$$

$$z, w \in \mathbb{Z} \tag{39}$$

where w are indicator variables and s is a set of slack variables corresponding to the non-satisfied coverage requirements.

Similarly to the KP, we can obtain an SAA formulation by sampling the coverage requirements d and introducing a set of slack variables for each scenario $\omega \in \Omega$:

$$\min \sum_{j \in \mathcal{J}} c_j z_j + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{i \in \mathcal{I}} \rho_{i,\omega} s_{i,\omega} \tag{40}$$

$$\sum_{j \in \mathcal{J}} a_{i,j} z_j \geq d_{i,\omega} (1 - w_{i,\omega}) \qquad \forall i \in \mathcal{I}, \omega \in \Omega$$

$$w_{i,\omega} = 1 \implies s_{i,\omega} \geq d_{i,\omega} - \sum_{j \in \mathcal{J}} a_{i,j} x_j \qquad \forall i \in \mathcal{I}, \omega \in \Omega$$

$$z_j \geq 0 \tag{41}$$

$$w_{i,\omega} \in [0, 1] \tag{42}$$

$$s_{i,\omega} \geq 0 \tag{43}$$

$$z, w \in \mathbb{Z} \tag{44}$$

C Methodological Details

Non-contextual standard deviation. While we employ stochastic parameter estimates, it is important to note that they are a part of our approach to smoothing and need not precisely reflect the actual distribution of y. Thus, we opt for a Gaussian distribution (defined through μ and σ), not because it most accurately represents the variance in y, but because it results in effective localized smoothing to obtain informative gradients. Throughout this research, we experimented with various ways of controlling the standard deviation σ, including the use of a constant hyperparameter σ, a trainable non-contextual σ (i.e., which is not dependent on the input features), and a contextual σ (i.e., which is input-dependent and is predicted alongside μ, based on features x). We observed that using a trainable non-contextual standard deviation generally produced the best tradeoff in terms of relative regret and number of epochs required for convergence (as shown in Figure 6). Introducing dependence on input features did not lead to a significant improvement, and thus, we use this setup in our experiments. This result intuitively makes sense, since σ only controls the degree of smoothing, and is not used to model any true variance that may be present in the data (after all, at test time we do not sample from the predicted distribution, and instead use point prediction $\mathbb{E}_{\hat{y} \sim p_\theta}[\hat{y}] = \mu$ directly).

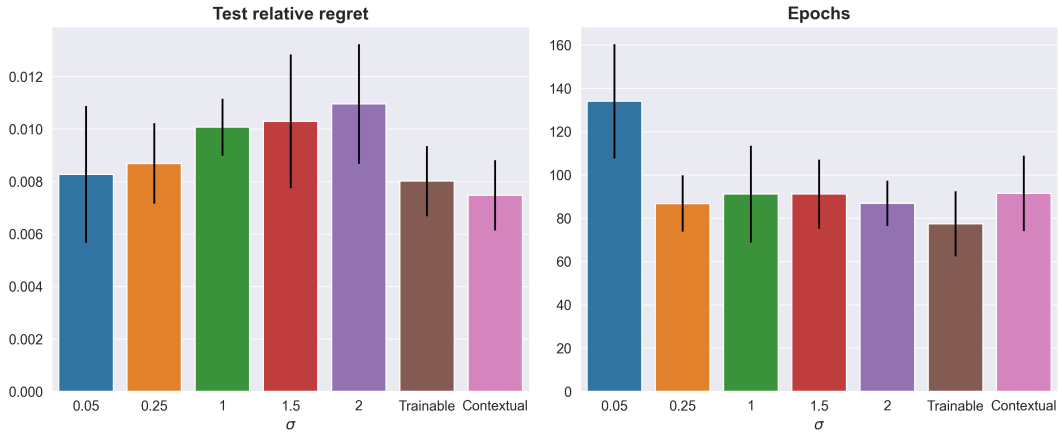

Fig. 6. Relative regret (left) and number of training epochs before early stopping (right) required by SFGE for different fixed values of σ, for a trainable σ and for a contextual σ, on the KP-50. We employed the same data generation and evaluation procedures described in Section 5.

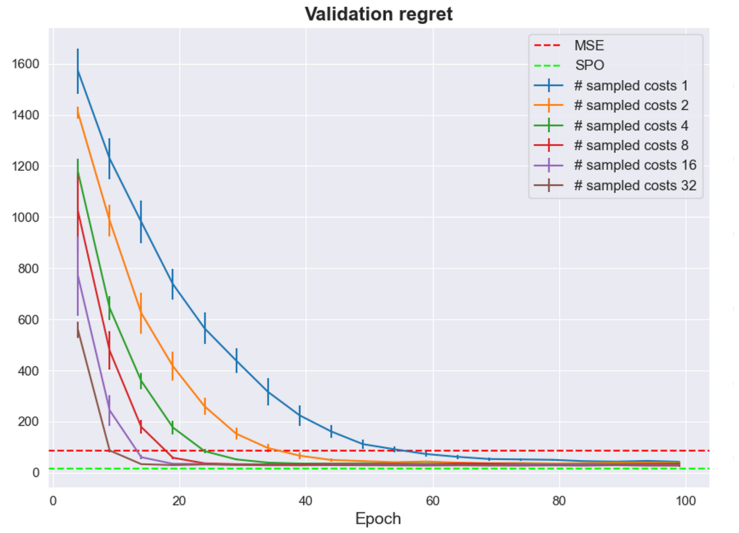

Fig. 7. The validation regret on the KP-50 with respect to the number of epochs, for different values of S (denoting the number of predictions $\hat{y}$ that are sampled per instance in each gradient estimation).

Number of samples. The gradient estimator is given by:

$$\nabla_\theta L(\theta, y) \approx \frac{1}{S} \sum_{i=1}^{S} \mathcal{L}(\hat{y}^{(i)}, y) \nabla_\theta \log p_\theta(\hat{y}^{(i)}) \tag{45}$$

The motivation for using a higher number of samples is to obtain a better estimate of the true gradient, which can improve convergence speed. However, this also requires solving more optimization problems per gradient

descent step, which may slow convergence speed in actual wall-clock time. We investigated this trade-off: as shown in Figure 7, using more samples results in fewer training epochs. However, we found that in terms of number of optimization problems solved until convergence (and consequently, wall clock time), $S = 1$ works best in our experiments. In general, the right value for S is likely to depend on the problem. For problems where the gradient estimate has high variance and the optimization problem can be solved relatively quickly, larger values for S may provide a better trade-off between the accuracy of the gradient estimate and the computational time required to compute it.

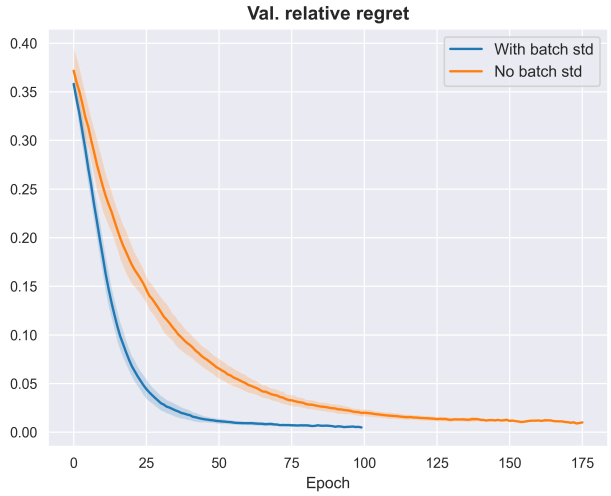

Fig. 8. Validation relative regret during training of SFGE with and without standardization on the KP-50

Variance reduction. As previously mentioned in the main body of the paper, we apply standardization to the regret within a single mini-batch to enhance convergence speed by reducing variance in the gradient estimation. The standardization is computed as follows:

$$\tilde{R} = \frac{R - \mu}{\sigma^2 + \epsilon}$$

Here, R represents the (post-hoc) regret, μ and σ denote the mean and variance of the regret within a mini-batch, and $\epsilon = 10^{-8}$ is a small constant introduced to prevent numerical instability. To empirically demonstrate the effectiveness of this standardization operation, we compare a model trained with SFGE with and without the standardization of the regret within mini-batches. We conducted experiments on the KP-50 dataset, following the same evaluation procedure as described in section 5. In Figure 8, we present a comparison of the validation relative regret between the two approaches, clearly illustrating that standardization significantly improves convergence speed.

Since the choice of mini-batch size affects the results of standardization and, consequently, the variance reduction, we conducted experiments with various batch sizes, specifically $\{2, 4, 8, 16, 32, 64, 128, 256, 512\}$, on the KP-50 dataset. We evaluated both the relative regret on the test set and the number of optimization problems solved before reaching convergence. The latter experiment provides insights into the computational efficiency of different configurations. As depicted in Figure 9, increasing the batch size results in a larger number of

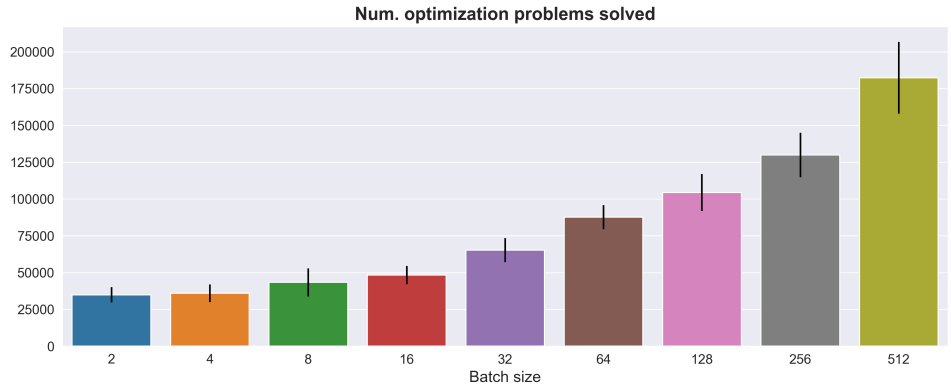

Fig. 9. Total number of optimization problems solved by SFGE during training w.r.t. the mini-batch size.

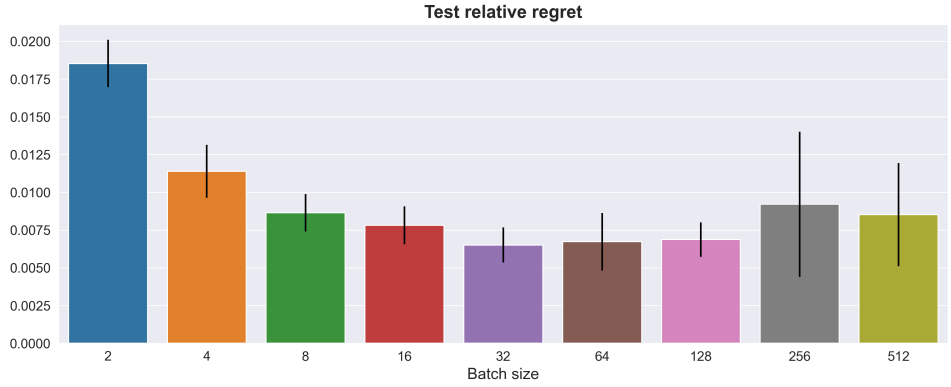

Fig. 10. SFGE test relative regret w.r.t. the mini-batch size.

optimization problems required to reach convergence, as it necessitates more epochs. With a larger batch size, the number of mini-batches decreases, reducing the number of optimization steps per epoch. Overall, a batch size of 32 demonstrates the best trade-off in terms of computational cost and relative regret.

D Comparison with Local-based Approximators in Stochastic Settings

In a stochastic setting, where y values are sampled from an unknown distribution depending on the contextual information x, such that $y \sim P(Y \mid x)$, regret itself is not deterministic anymore and its expected value on a given point x_i can be expressed as:

$$\mathbb{E}[R_{x_i}] = \int p(y, x_i) Regret(m_\omega(x_i), y) dy \tag{46}$$

Where m is a predictive model parameterized on ω. This implies that the expected global minima for the regret loss function with respect to predictions may differ from the observed ones. Most importantly, a correct method

capable of handling the stochastic scenario would have to converge to the expected minimum, assuming to have enough data.

In this section, we will show by means of a counterexample how methods based on local approximations, mainly represented by the work of Shah et al. (2022a), which will be referred to as *LODL*, fail at giving such a guarantee, in contrast with our approach.

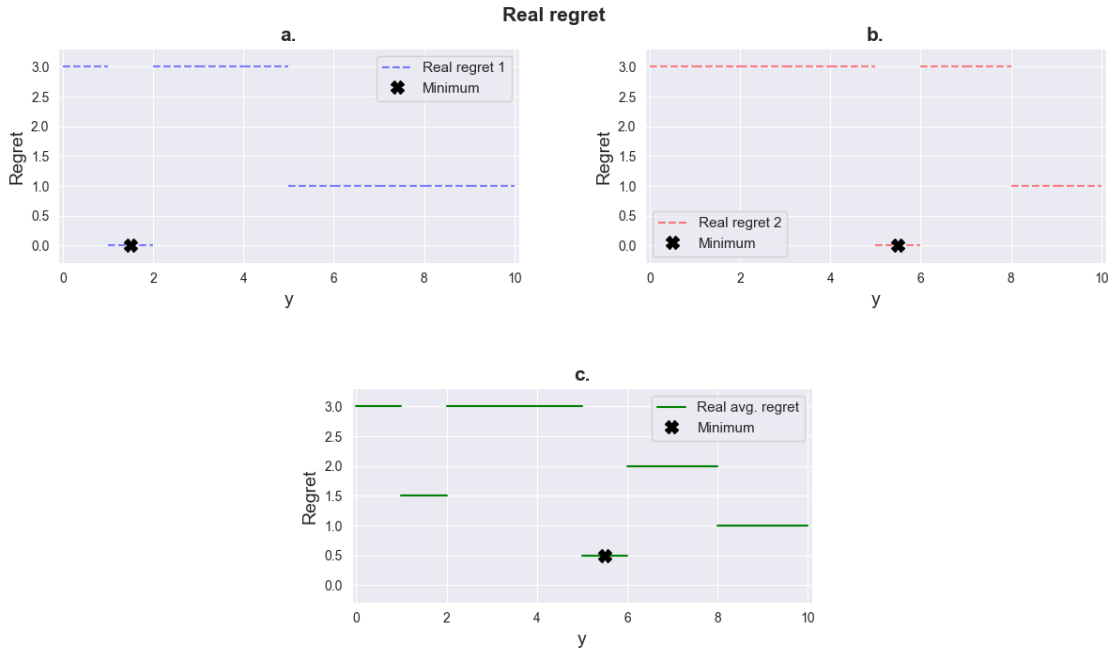

Fig. 11. Regret functions on a fixed x_i for varying values of y. **a.** Depicts regret r_1 for $\hat{y} = y_1$ (first scenario). **b.** Depicts regret r_2 for $\hat{y} = y_2$ (second scenario). **c.** Depicts the average regret between r_1 and r_2.

As a simple example, assume to have an instance of a problem, where x_i is fixed and $y \in \mathbb{R}$, with only two possible scenarios:

(1) $y_1 \in [1.0, 2.0]$, with probability $p(y_1, x_i) = 0.5$
(2) $y_2 \in [5.0, 6.0]$, with equal probability $p(y_2, x_i) = 0.5$

These two possibilities determine two distinct regrets r_1 and r_2, with one loss landscape being equal to the other, except for a translation along the x-axis, as shown in Figure 11 (**a.** and **b.**). Each of them showcases a different global minimum position. The expected regret (**c.**) is the average of the two (since y is sampled from a uniform distribution) and it is characterized by a different minimum, coinciding with the one of r_2, as depicted in Figure 11.

Given a sufficient number of data points, the expected regret identified by our method is obtained by applying the smoothing procedure to both r_1 and r_2 and then averaging. Since the smoothing is a global approximation of the original functions, the resulting average is itself a global (smoothed) approximation of the real average regret, guaranteeing the correct location of the optima, up to a certain level of precision, depending on the smoothing

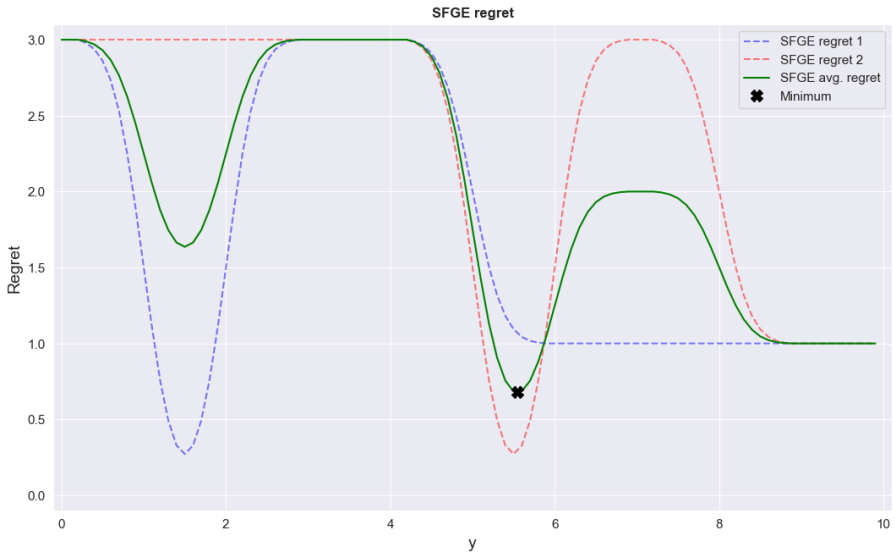

Fig. 12. Surrogate regret functions obtained by smoothing via SFGE on r_1 (blue line) and r_2 (red line). The minimum of their average (green line) matches with the expected one.

strength (e.g., a strong one - with a high value of σ - may cause excessive flattening and loss of information). Figure 12 shows regret approximations based on this method. The minimum is correctly located at $y = 5.5$.

LODL, on the other hand, cannot do the same, as the average of its local approximations does not guarantee to produce a good local approximation of the expected regret around the expected minimum. Figure 13 shows how, by fitting two quadratic models around global minima of r_1 and r_2, their average (another quadratic function, by construction) is still convex, but it misplaces the expected global minimum, which is moved to $y = 3.5$.

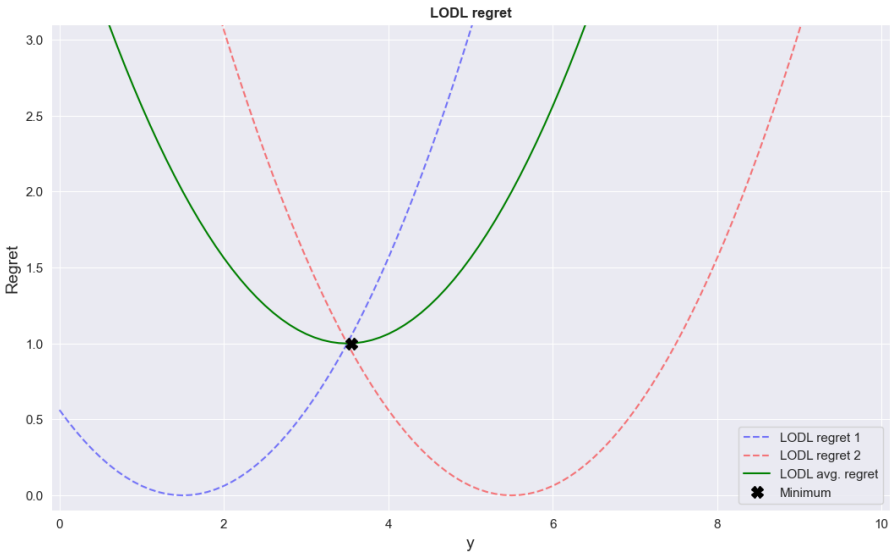

Fig. 13. Surrogate regret functions obtained by local approximations on r_1 (blue line) and r_2 (red line). The minimum of their average (green line) is far away from the expected one.

E Behavior of the Learned Standard Deviation During Training

In this section, we examine the behavior of the learned standard deviation σ of the parameterized Gaussian distribution during training with SFGE. For clarity of visualization, we report results on the knapsack problem with 10 and 5 items, where the cost coefficients are unknown.

As shown in Figure 14, σ generally decreases over the course of training, though not always monotonically. The decay pattern also varies across cost coefficients. For instance, in KP-10 (Figure 14a), σ gradually decreases for some coefficients, while, for the others, it first exhibits a slight increase during the initial ≈ 10 epochs before transitioning into a gradual decay. This might be the result of an initial exploratory phase, which requires a high level of smoothing to obtain more informative gradients. As training proceeds, σ enters a more stable decay, providing a lower level of smoothness and, consequently, the smoothed loss becomes more similar to the original one (the post-hoc regret), thus preserving the true optimum. In the KP-5, the decay is more pronounced than in KP-10, illustrating that the learning dynamics are problem-dependent. This highlights the advantage of using a trainable σ: the optimizer automatically adjusts σ without requiring a predefined decay schedule. Furthermore, different dimensions of the predicted vector may benefit from distinct decay behaviors, which are naturally captured by this approach.

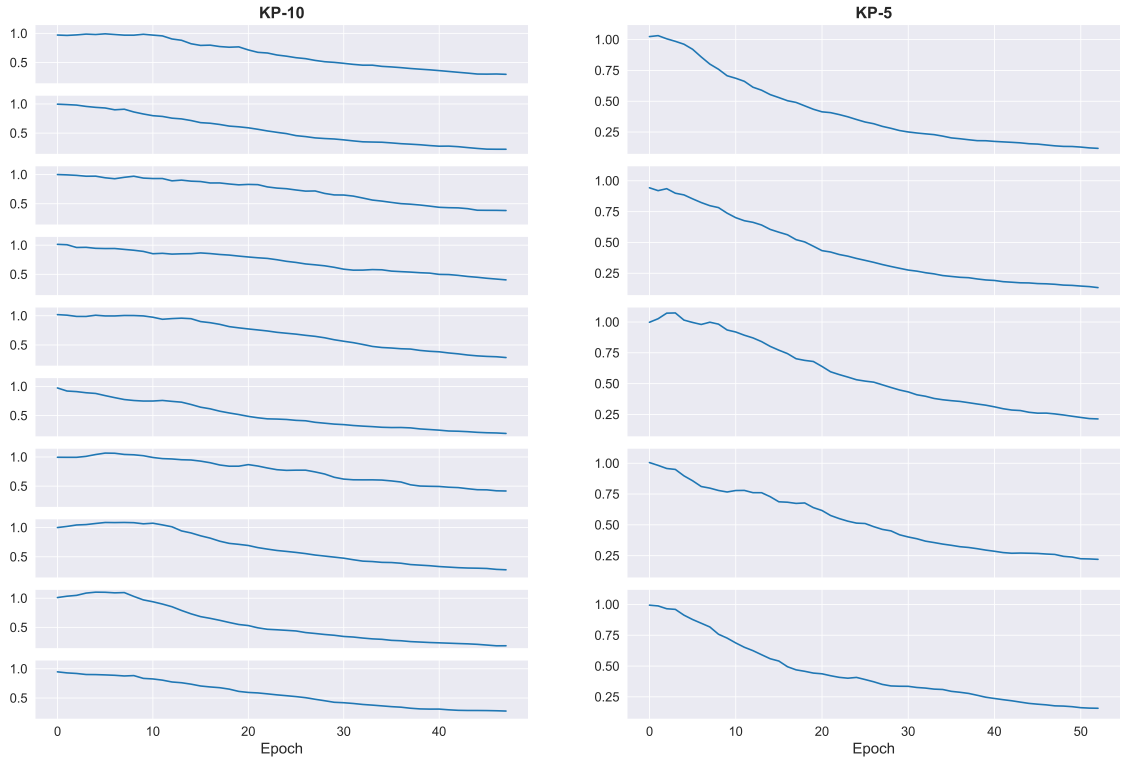

Fig. 14. The value of the standard deviation of the parametrized Gaussian distribution trained with SFGE w.r.t. the training epoch on the KP-10 (a) and KP-5 (b).

F Relationship among the SFGE Performance and the Predicted Parameters Size

In this section, we investigate the performance of SFGE with respect to the predicted parameters size. We consider the KP problem of sizes 50, 75, 100 and 200, with unknown cost coefficients. The dataset are generated with the same procedure described in section 5. We compare PFL, SPO and SFGE-MAP in terms of relative regret and number of epochs required to reach convergence. The hyperparameters are the same described in section 5.

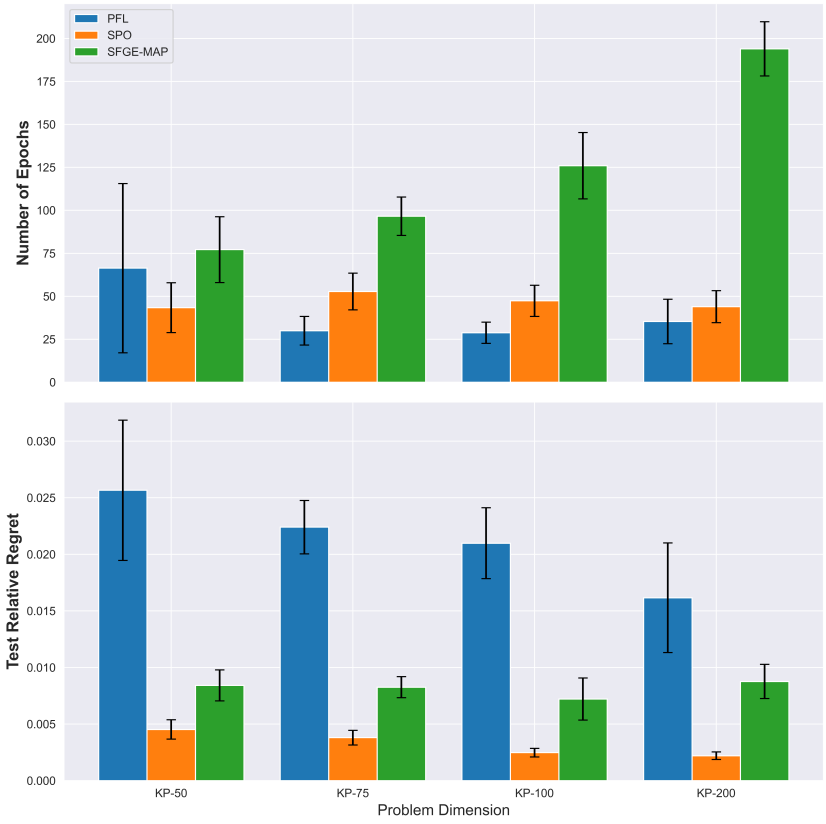

Fig. 15. Relative regret and number of epochs required to reach convergence w.r.t. the KP problem size.

Results are shown in Figure 15. While SFGE maintains stable relative regret across different parameter vector sizes, its convergence speed degrades as dimensionality increases, requiring more epochs for convergence.

G Additional Results

In this section, we present supplementary results that further validate and extend the conclusions drawn in the main paper across different problem dimensions and specifications. Specifically, we report results for the following settings: 1) Prediction of the item values for the KP with 75 items and quadratic KP with 8 and 10 items (Table 6); 2) Prediction of the item values and weights for the fractional KP with capacity value of 75 (Table 7); 3) prediction of the capacity value for the KP with unknown capacity value with 50 items (Table 8); 4) prediction of the coverage requirement for the WSMC of size 5×25 (Table 9); 5) prediction of weights for KP with 50 items

and stochastic weights, testing on multiple (30) samples for each input (Table 10): these experiments further demonstrate the ability of SFGE to converge to a better expected estimation, compared to the other approaches.

Table 6. PFL, SFGE-MAP and SPO results on the linear and quadratic KP.

Method	*Rel. regret*	*MSE* ($\times 10^4$)	*Epochs*
	KP-50		
PFL	0.022 ± 0.005	**2.38 ± 1.28**	**40.5 ± 28.5**
SPO	0.003 ± 0.0004	4.12 ± 1.52	**40.3 ± 9.77**
DPO	**0.002 ± 0.001**	6.24 ± 3.30	45.0 ± 13.4
SFGE	0.010 ± 0.002	17.8 ± 4.42	141 ± 14.0
SFGE-MAP	0.008 ± 0.002	10.7 ± 2.03	83.1 ± 8.03
SFGE-MAP (contextual std.dev)	0.008 ± 0.002	9.98 ± 2.31	89.3 ± 13.9
	KP-75		
PFL	0.021 ± 0.006	**2.15 ± 1.11**	44.9 ± 35.3
SPO	**0.003 ± 0.001**	4.24 ± 1.18	51.5 ± 10.9
DPO	**0.002 ± 0.0004**	5.17 ± 2.32	45.6 ± 7.00
SFGE-MAP	0.008 ± 0.001	12.6 ± 4.22	103.8 ± 14.7
	Quadratic KP-8		
PFL	0.034 ± 0.015	**2.35 ± 1.58**	**24.3 ± 4.61**
SFGE-MAP	0.006 ± 0.003	10.6 ± 6.45	54.5 ± 14.9
SPO	**0.005 ± 0.002**	6.95 ± 4.09	29.9 ± 10.4
	Quadratic KP-10		
PFL	0.041 ± 0.011	**2.37 ± 1.50**	45.8 ± 64.0
SFGE-MAP	0.008 ± 0.002	8.46 ± 4.04	54.1 ± 13.1
SPO	**0.006 ± 0.002**	6.47 ± 3.26	30.7 ± 8.6

Table 7. PFL, SFGE and PO results on the fractional KP of different sizes and for different penalty coefficient values.

Method	*Rel. PRegret*	*Feas. rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
capacity=75, $\rho = 0$					
PFL	0.353 ± 0.014	**0.096 ± 0.058**	**0.72 ± 0.15**	**99.1 ± 13.1**	13.3 ± 2.9
SFGE	0.337 ± 0.009	–	0.99 ± 0.01	$6.20 \cdot 10^5 \pm 6.06 \cdot 10^5$	13.4 ± 4.1
P+O	0.332 ± 0.109	–	1.0 ± 0.0	$9.8 \cdot 10^5 \pm 1.1 \cdot 10^4$	**2.4 ± 1.4**
capacity=75, $\rho = 1$					
PFL	0.437 ± 0.023	**0.096 ± 0.058**	0.72 ± 0.15	**99.1 ± 13.1**	13.3 ± 2.87
SFGE	0.410 ± 0.010	0.172 ± 0.044	**0.52 ± 0.09**	$9.41 \cdot 10^5 \pm 5.40 \cdot 10^5$	16.3 ± 3.8
P+O	0.405 ± 0.145	0.332 ± 0.013	0.617 ± 0.035	$3.8 \cdot 10^5 \pm 4.5 \cdot 10^3$	**1.8 ± 1.5**
capacity=75, $\rho = 2$					
PFL	0.522 ± 0.057	**0.096 ± 0.058**	0.72 ± 0.15	**99.1 ± 13.1**	13.3 ± 2.87
SFGE	0.436 ± 0.010	0.230 ± 0.081	**0.40 ± 0.16**	$2.1 \cdot 10^6 \pm 1.53 \cdot 10^6$	20.8 ± 7.8
P+O	0.426 ± 0.149	0.378 ± 0.009	0.428 ± 0.025	$3.5 \cdot 10^5 \pm 4.0 \cdot 10^3$	**3.2 ± 2.0**

Table 8. PFL and SFGE results on the KP with uncertain capacity of different sizes and for different penalty coefficient values.

Method	*Rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
50-items, $\rho = 5$				
PFL	0.556 ± 0.353	**0.667 ± 0.257**	**10.81 ± 6.30**	**11.5 ± 1.9**
SFGE	**0.367 ± 0.238**	0.752 ± 0.298	16.82 ± 12.00	44.9 ± 14.5
50-items, $\rho = 10$				
PFL	1.213 ± 0.780	**0.667 ± 0.257**	**10.81 ± 6.30**	**11.5 ± 1.9**
SFGE	**0.556 ± 0.322**	0.893 ± 0.076	23.24 ± 16.78	57.2 ± 30.4
50-items, $\rho = 20$				
PFL	2.520 ± 1.631	**0.667 ± 0.257**	**10.812 ± 6.295**	**11.5 ± 1.9**
SFGE	**0.800 ± 0.466**	0.953 ± 0.031	32.415 ± 22.131	48.1 ± 21.1

Table 9. PFL and SFGE results on the WSMC.

Method	*Rel. PRegret*	*Feas. rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
5×25, $\rho = 1$					
PFL	1.18 ± 0.62	$\mathbf{0.154 \pm 0.076}$	0.63 ± 0.11	$\mathbf{4.74 \cdot 10^4 \pm 2.10 \cdot 10^4}$	$\mathbf{42.8 \pm 23.1}$
CombOptNet	4.05 ± 2.58	–	0.97 ± 0.04	$6.20 \cdot 10^6 \pm 5.24 \cdot 10^6$	57.4 ± 30.8
SFGE	$\mathbf{0.850 \pm 0.264}$	0.314 ± 0.361	$\mathbf{0.55 \pm 0.27}$	$1.80 \cdot 10^5 \pm 6.85 \cdot 10^4$	55.2 ± 15.5
5×25, $\rho = 5$					
PFL	7.27 ± 4.20	$\mathbf{0.182 \pm 0.037}$	0.60 ± 0.11	$\mathbf{8.03 \cdot 10^4 \pm 2.08 \cdot 10^4}$	49.5 ± 24.2
CombOptNet	149.76 ± 91.56	–	0.99 ± 0.02	$9.08 \cdot 10^6 \pm 3.87 \cdot 10^6$	$\mathbf{40.1 \pm 19.1}$
SFGE	$\mathbf{2.53 \pm 0.53}$	1.28 ± 0.44	$\mathbf{0.23 \pm 0.11}$	$4.70 \cdot 10^5 \pm 1.94 \cdot 10^5$	48.8 ± 10.2
5×25, $\rho = 10$					
PFL	16.0 ± 10.1	$\mathbf{0.12 \pm 0.03}$	0.67 ± 0.07	$\mathbf{7.55 \cdot 10^4 \pm 5.65 \cdot 10^4}$	$\mathbf{37.9 \pm 16.3}$
CombOptNet	602.1 ± 276.7	–	0.98 ± 0.06	$1.27 \cdot 10^7 \pm 1.55 \cdot 10^7$	47.6 ± 26.5
SFGE	$\mathbf{3.00 \pm 0.50}$	1.72 ± 0.41	$\mathbf{0.12 \pm 0.07}$	$4.19 \cdot 10^5 \pm 1.06 \cdot 10^5$	47.5 ± 12.8

Table 10. PFL and SFGE results on the KP-50 with uncertain weights and 30 y samples for each x. We omit the *Feas. rel. PRegret* for *Infeas. ratios* near 1.

Method	*Rel. PRegret*	*Feas. rel. PRegret*	*Infeas. ratio*	*MSE*	*Epochs*
50-items, $\rho = 5$					
PFL	0.614 ± 0.128	0.001 ± 0.001	0.93 ± 0.03	$\mathbf{1.73 \cdot 10^5 \pm 3.23 \cdot 10^4}$	$\mathbf{34.0 \pm 20.3}$
CombOptNet	0.719 ± 0.244	0.008 ± 0.005	0.91 ± 0.02	$2.67 \cdot 10^8 \pm 8.02 \cdot 10^7$	41.0 ± 13.4
SFGE(ours)	$\mathbf{0.374 \pm 0.061}$	-	0.94 ± 0.02	$2.88 \cdot 10^5 \pm 8.79 \cdot 10^4$	136.0 ± 10.8
50-items, $\rho = 10$					
PFL	1.218 ± 0.242	0.001 ± 0.001	0.93 ± 0.03	$\mathbf{1.73 \cdot 10^5 \pm 3.23 \cdot 10^4}$	$\mathbf{34.0 \pm 20.3}$
CombOptNet	1.685 ± 0.501	0.008 ± 0.005	0.91 ± 0.02	$2.67 \cdot 10^8 \pm 8.02 \cdot 10^7$	41.0 ± 13.4
SFGE(ours)	$\mathbf{0.736 \pm 0.103}$	-	0.94 ± 0.02	$3.06 \cdot 10^5 \pm 9.23 \cdot 10^4$	128.8 ± 19.2
50-items, $\rho = 20$					
PFL	2.418 ± 0.481	0.001 ± 0.001	0.93 ± 0.03	$\mathbf{1.73 \cdot 10^5 \pm 3.23 \cdot 10^4}$	$\mathbf{34.0 \pm 20.3}$
CombOptNet	3.178 ± 0.989	0.008 ± 0.005	0.91 ± 0.02	$2.67 \cdot 10^8 \pm 8.02 \cdot 10^7$	41.0 ± 13.4
SFGE(ours)	$\mathbf{1.106 \pm 0.164}$	-	0.94 ± 0.02	$3.12 \cdot 10^5 \pm 9.52 \cdot 10^4$	119.3 ± 26.1

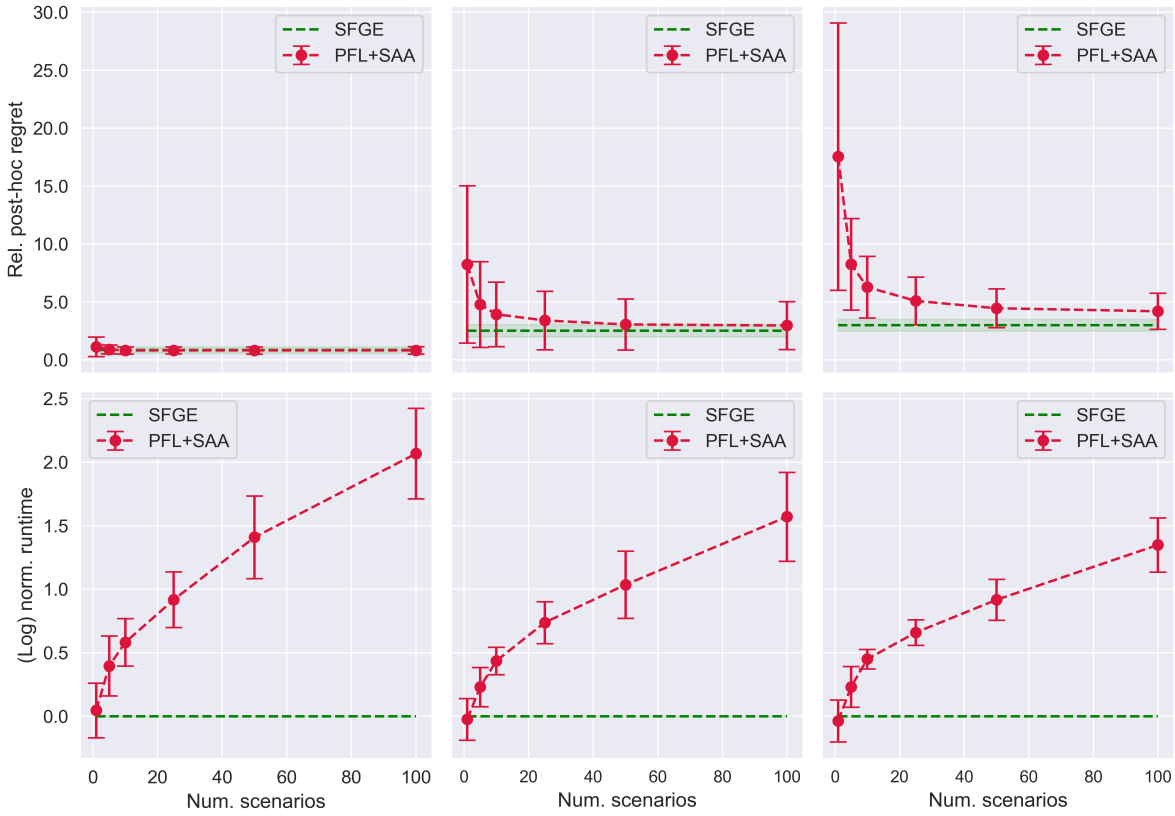

Fig. 16. Comparison between SFGE and PFL+SAA on the WSMC of size 5×25 for $\rho = 1$ (left), $\rho = 5$ (center) and $\rho = 10$ (right).

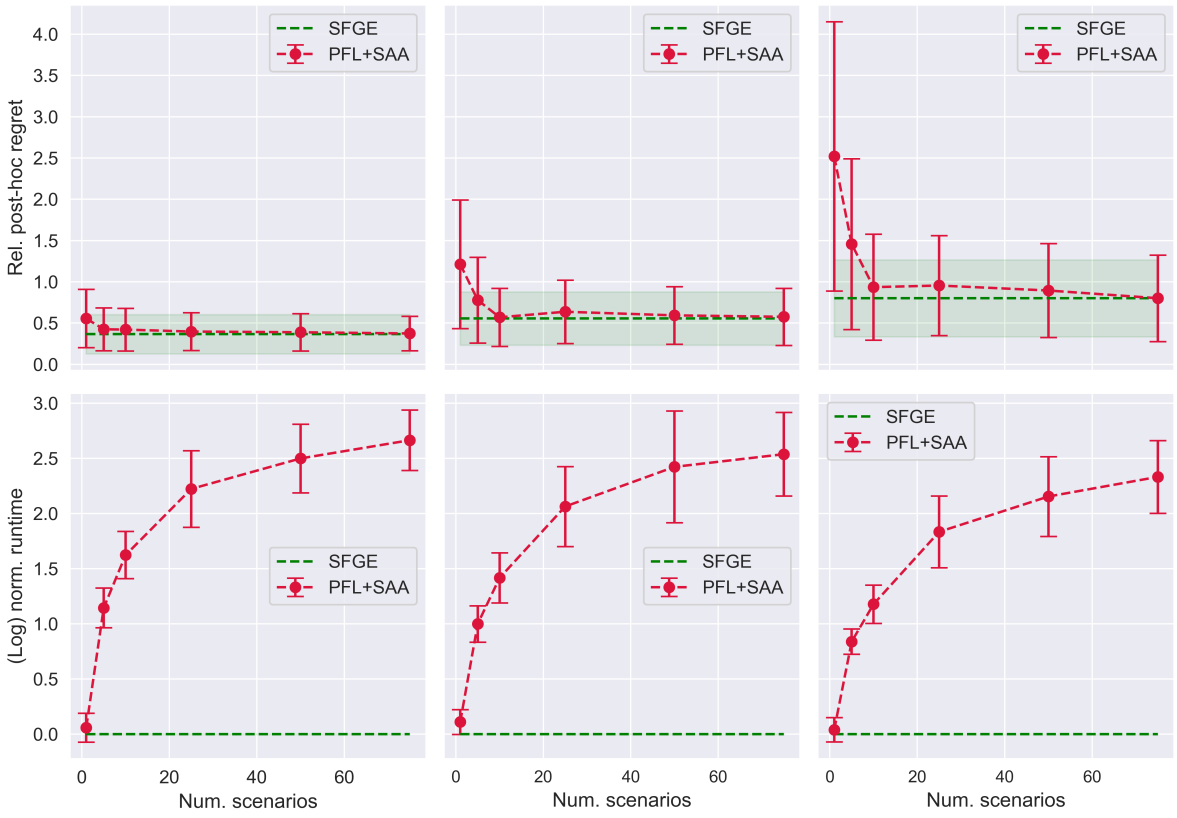

Fig. 17. Comparison between SFGE and PFL+SAA on the KP-50 with stochastic capacity, for $\rho = 5$ (left), $\rho = 10$ (center) and $\rho = 20$ (right).

Received 11 June 2025; accepted 1 December 2025