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

Mattia Silvestri [1] **Senne Berden** [2] **Jayanta Mandi** [2] **Ali İrfan Mahmutoğulları** [2] **Maxime Mulamba** [2]
Allegra De Filippo [1] **Tias Guns** [2] **Michele Lombardi** [1]

Abstract

Many real-world optimization problems contain unknown parameters that must be predicted prior to solving. To train the predictive machine learning (ML) models involved, the commonly adopted approach focuses on maximizing predictive accuracy. However, this approach does not always lead to the minimization of the downstream task loss. Decision-focused learning (DFL) is a recently proposed paradigm whose goal is to train the ML model by directly minimizing the task loss. However, state-of-the-art DFL methods are limited by the assumptions they make about the structure of the optimization problem (e.g., that the problem is linear) and by the fact that can only predict parameters that appear in the objective function. In this work, we address these limitations by instead predicting *distributions* over parameters and adopting score function gradient estimation (SFGE) to compute decision-focused updates to the predictive model, thereby widening the applicability of DFL. Our experiments show that by using SFGE we can: (1) deal with predictions that occur both in the objective function and in the constraints; and (2) effectively tackle two-stage stochastic optimization problems.

1. Introduction

Many real-world decision-making problems contain parameters that are uncertain at solving time. Consider, for example, a manufacturing company that needs to schedule its production in function of uncertain customer demands, or a delivery company that needs to route its vehicles in function of uncertain traffic conditions. These kinds of problems can be framed as *predict-then-optimize problems*. As indicated by their name, predict-then-optimize problems consist of two stages – a prediction stage and an optimization stage. In the prediction stage, a machine learning (ML) model is used to predict the unknown quantities. In the optimization stage, the predicted quantities occur as parameters in an optimization problem, which is solved by optimizing some objective function while satisfying a set of domain-specific constraints. Effective solving of predict-then-optimize problems hinges on the quality of the ML model used, and thus the way this model is trained is highly important. Two paradigms for doing so can be distinguished: prediction-focused and decision-focused learning.

In *prediction-focused learning* (PFL) (also called two-stage learning in (Wilder et al., 2019)), the predictive model is trained without considering the downstream optimization problem. In other words, it is trained to maximize the accuracy of the predicted parameters, using traditional ML losses like the mean squared error (MSE). While this may appear entirely sensible at first – after all, higher predictive accuracy generally leads to better decisions – it does not take into account the complex ways in which prediction errors affect the downstream decision-making.

To account for this, *decision-focused learning* (DFL) trains the predictive model to directly minimize the task loss at hand, i.e., some metric that depends on the outcome of the optimization problem instantiated by the predicted parameters. This involves a deeper integration of the prediction and optimization phases, as training the predictive model now requires backpropagation through the optimization procedure. While this can be done exactly for convex optimization problems through implicit differentiation (Agrawal et al., 2019; Amos & Kolter, 2017), it is more tricky when the optimization problem is combinatorial. This is because, when the parameters of a combinatorial optimization problem change, the solution either does not change at all, or changes discontinuously. In other words, the partial derivatives of the solution with respect to the parameters are zero almost everywhere, and do not exist at the points where the solution changes suddenly. This makes the straightforward

[*]Equal contribution [1]Dipartimento di Informatica Scienza e Ingegneria, University of Bologna, Bologna, Italy [2]Department of Computer Science, KU Leuven, Belgium. Correspondence to: Mattia Silvestri <mattia.silvestri4@unibo.it>.

Published at the Differentiable Almost Everything Workshop of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023. Copyright 2023 by the author(s).

application of gradient descent unhelpful in training.

To address this challenge, previous works (Elmachtoub & Grigas, 2022; Elmachtoub et al., 2020; Mandi & Guns, 2020; Mandi et al., 2022; Mulamba et al., 2021; Shah et al., 2022; Wilder et al., 2019) have proposed different techniques to still obtain useful gradients. However, most works consider optimization problems where the predicted parameters appear only in the objective function. This is a major limitation, as in many real-world problems, both the objective and constraint functions contain uncertain parameters. Moreover, most methods are restricted to problems in which the objective function is linear. These limitations in the state of the art motivated us to develop a more widely applicable methodology for DFL. More concretely, this paper investigates the use of score function gradient estimation (i.e., the REINFORCE algorithm (Williams, 1992)) to learn to predict any parameters of an optimization problem (in the objective, in the constraints, or both), regardless of whether the objective function is linear or not. We also show how, thanks to its generality, the method can effectively tackle stochastic optimization problems, with consistent improvement over PFL approaches.

2. Problem setting

We consider a parametric optimization problem:

$$z^\star(y) = \arg\min_z \; f(z, y) \tag{1a}$$

$$\text{s.t.} \;\; g(z, y) \leq 0 \tag{1b}$$

$$h(z, y) = 0 \tag{1c}$$

The goal of the optimization problem is to find a solution $z^\star(y)$, a minimizer of the objective function f, that satisfies a set of inequality and equality constraints defined by g and h, respectively. Parameter vector y is unknown, but can be estimated as a function of some correlated known features x. Given is a set of examples $D = \{(x_i, y_i)\}_{i=1}^N$. This is used to train a machine learning model m_ω that makes predictions $\hat{y} = m_\omega(x)$.

In the predict-then-optimize setting, the goal in training the predictive model is not to maximize the accuracy of $\hat{y}$ with respect to y. Rather, the goal is to learn to make predictions $\hat{y}$ that lead to good decisions $z^\star(\hat{y})$, with respect to some task loss $\mathcal{L}$. A common task loss is the *regret* (also called the SPO loss in (Elmachtoub & Grigas, 2022)), which expresses the suboptimality of the decisions made on the basis of the predicted parameters, with respect to the ground-truth parameters:

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

The regret is commonly used because most DFL work is limited to predicting parameters of the objective function. The regret, however, is not appropriate when the predicted parameters also occur in the constraints. To this end, (Hu et al., 2022) introduced the notion of *post-hoc regret*, which is based on the use of a correction and a penalty function. The correction function turns a solution $z^\star(\hat{y})$ that is infeasible with respect to y into a feasible one $z^\star_{corr}(\hat{y}, y)$, while the penalty function $Pen(z^\star(\hat{y}), z^\star_{corr}(\hat{y}, y))$ expresses the cost for doing so. Note that feasible solutions are mapped onto themselves by the correction function, and that the associated penalty for this is 0. The post-hoc regret $PRegret$ captures the suboptimality of the corrected decisions, plus the associated penalty:

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

Note that while we consider the post-hoc regret as task loss in the experiments, our method does not depend on it and will be formalized in function of an arbitrary task loss.

3. Score Function Gradient Estimation

The central problem in decision-focused learning is that when the loss $\mathcal{L}$ depends on the outcome $z^\star$ of a combinatorial optimization procedure, it has zero-valued gradients with respect to the predictive model's parameters almost everywhere. This can be seen when applying the chain rule:

$$\frac{\partial \mathcal{L}(z^\star(\hat{y}), y)}{\partial \omega} = \frac{\partial \mathcal{L}(z^\star(\hat{y}), y)}{\partial z^\star(\hat{y})} \frac{\partial z^\star(\hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \omega} \tag{4}$$

The second factor, $\frac{\partial z^\star(\hat{y})}{\partial \hat{y}}$, measures the change in $z^\star(\hat{y})$ when $\hat{y}$ changes infinitesimally. However, since the problem is combinatorial, this change is zero almost everywhere. This in turn causes the entire gradient $\frac{\partial \mathcal{L}(z^\star(\hat{y}), y)}{\partial \omega}$ to be zero almost everywhere, and thus to be unhelpful in gradient-based learning.

To tackle this, we shift from training a model that makes point predictions $\hat{y}$, to a model that predicts a vector θ that instantiates a distribution $p_\theta(y)$. In other words, instead of predicting parameter vectors, the model predicts *distributions* over parameter vectors. For instance, we consider y to be sampled from a multivariate Gaussian distribution parameterized by its means μ and standard deviations σ, which the predictive model is trained to predict, i.e., $\theta = (\mu, \sigma)$. The loss then becomes an expectation:

$$L(\theta, y) = \mathbb{E}_{\hat{y} \sim p_\theta(y)}[\mathcal{L}(\hat{y}, y)] \tag{5}$$

The motivation for this is that it removes the zero-gradient problem: by predicting distributions, the gradient of the loss with respect to the output of the predictive model is not zero anymore. Despite overcoming differentiability via stochastic smoothing is not novel (Niepert et al., 2021), (Petersen,

2022), (Berthet et al., 2020), (Abernethy et al., 2016), to the best our knowledge, this is the first time it is applied in the context of DFL. However, computing the gradient of the loss is not trivial. To do so, we take inspiration from the field of reinforcement learning, where score function gradient estimation (i.e., the REINFORCE algorithm) is used. Consider the following derivation:

$$\nabla_\theta L(\theta, y) = \nabla_\theta \mathbb{E}_{\hat{y}\sim p_\theta(y)}[\mathcal{L}(\hat{y}, y)] \tag{6a}$$

$$= \nabla_\theta \int p_\theta(y)\mathcal{L}(\hat{y}, y)d\hat{y} \tag{6b}$$

$$= \int \mathcal{L}(\hat{y}, y)\nabla_\theta p_\theta(\hat{y})d\hat{y} \tag{6c}$$

$$= \int \mathcal{L}(\hat{y}, y)p_\theta(\hat{y})\nabla_\theta \log p_\theta(\hat{y})d\hat{y} \tag{6d}$$

$$= \mathbb{E}_{\hat{y}\sim p_\theta(y)}[\mathcal{L}(\hat{y}, y)\nabla_\theta \log p_\theta(\hat{y})] \tag{6e}$$

The validity of bringing the gradient inward in line 6c is discussed in (Mohamed et al., 2020). In line 6d, the log derivative trick is used.

To get an estimation of this gradient that can effectively be used in training, the final result in line 6e can be estimated using a Monte Carlo method, giving

$$\nabla_\theta L(\theta, y) \approx \frac{1}{S}\sum_{i=1}^{S} \mathcal{L}(\hat{y}^{(i)}, y)\nabla_\theta \log p_\theta(\hat{y}^{(i)}) \quad \text{with } \hat{y}^{(i)} \sim p_\theta(y) \tag{7}$$

with S as the total number of samples.

Note that the method is not specific to any loss function $\mathcal{L}$. For example, this can be the Hamming distance between solutions $z^\star(\hat{y})$ and $z^\star(y)$, the regret, or the post-hoc regret. In our experiments, the latter is used.

4. Experimental Results

We conducted a preliminary analysis by focusing on two main research questions: 1) *Does* SFGE *outperform PFL when predicting parameters that appear in the constraints?*; 2) *Does* SFGE *outperform PFL when predicting parameters of stochastic optimization problems?*

Prediction of parameters in the constraints To the best of our knowledge, the only work that tackles the problem of predicting parameters in the constraints in a decision-focused manner is (Hu et al., 2022) but it is limited to linear packing and covering problems, and thus is not applicable to general integer linear programs. A wider applicable approach is proposed in (Paulus et al., 2021); however, the machine learning model is trained end-to-end to provide a more accurate solution rather than making decision-focused prediction.

We consider the knapsack problem (KP) wherein the item weights are unknown. Since the predicted weights may lead to a solution that exceeds the capacity when considering the ground-truth weights, a correction action is required to recover feasibility. In the case of the KP, this function discards a subset of the chosen items, for which an additional cost is payed, whose value depends on a penalty coefficient ρ. More precisely, for the i-th item, the cost for discarding it is $\rho \cdot v_i$, where v_i is the item value.

Table 1. PFL and SFGE results on the KP with uncertain weights of size 50 for different penalty coefficient values.

Method	*Rel. PRegret*	*Infeas. ratio*
	$\rho = 1$	
PFL	0.119 ± 0.014	0.33 ± 0.05
SFGE	$\mathbf{0.103 \pm 0.019}$	$\mathbf{0.23 \pm 0.06}$
	$\rho = 5$	
PFL	0.259 ± 0.046	0.33 ± 0.05
SFGE	$\mathbf{0.187 \pm 0.069}$	$\mathbf{0.05 \pm 0.03}$
	$\rho = 10$	
PFL	0.435 ± 0.091	0.33 ± 0.05
SFGE	$\mathbf{0.235 \pm 0.078}$	$\mathbf{0.03 \pm 0.03}$

We compare SFGE with a PFL approach that trains a ML model to minimize the MSE. For each of the methods, we report the relative post-hoc regret (*Rel. PRegret*) and the ratio of infeasible solutions (*Infeas. ratio*), which expresses how frequently the recourse action is used. We run experiments on three different KPs, all with 50 items , but with differing penalty coefficients ρ, namely $\rho = \{1, 5, 10\}$. We generated synthetic data by introducing a mapping between input features and targets in the same way as described in the shortest path experimental evaluation of (Elmachtoub & Grigas, 2022), with a degree of model misspecification $deg = 5$ and number, $p = 5$, of input features, and a noise half-width $\bar{\epsilon} = 0.5$. The results are shown in Table 1: SFGE *outperforms* PFL *in all the cases and it provides both better relative regret and infeasibility ratio*.

Stochastic Optimization To assess the performance of SFGE on stochastic optimization problems, we run experiments on the weighted set multi-cover problem (WSMC) with stochastic coverage requirements. A recourse action is employed whenever the demands are not satisfied at the price of paying an additional cost whose value depends on a penalty coefficient. More specifically, the additional cost for each unit of not satisfied demand of the i-th product is computed as the maximum set cost among the ones that cover it, multiplied by the coefficient ρ. We conducted the experimental analysis on synthetic data that were generated following a set of guidelines that allows us to obtain realistic instances (Grossman & Wool, 1997). The set costs

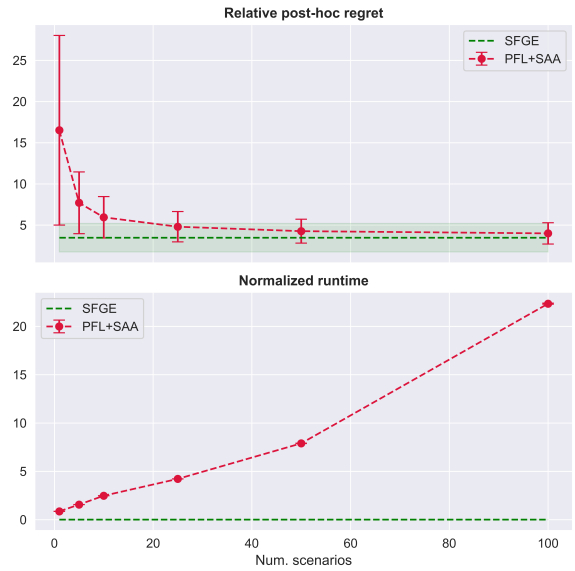

Figure 1. The relative post-hoc regret and normalized runtime at inference time of SFGE and PFL+SAA on the WSMC

are randomly generated in the range $[1, 100]$ with a uniform probability distribution. Finally, we assume that the demands follow a Poisson distribution whose rate λ depends on a set of correlated features. The ground-truth mapping between features and rate is the same as the one previously described for the KP, and was taken from (Elmachtoub & Grigas, 2022).

Table 2. PFL and SFGE results on the WSMC with 5 items and 25 sets for different penalty coefficient values.

Method	*Rel. PRegret*	*Infeas. ratio*
	$\rho = 1$	
PFL	1.18 ± 0.62	0.63 ± 0.11
SFGE	$\mathbf{0.850 \pm 0.264}$	$\mathbf{0.55 \pm 0.27}$
	$\rho = 5$	
PFL	7.27 ± 4.20	0.63 ± 0.11
SFGE	$\mathbf{2.53 \pm 0.53}$	$\mathbf{0.23 \pm 0.11}$
	$\rho = 10$	
PFL	17.8 ± 13.3	0.63 ± 0.11
SFGE	$\mathbf{3.47 \pm 1.73}$	$\mathbf{0.12 \pm 0.06}$

To the best of our knowledge, the only work that proposes a method to tackle stochastic optimization problems in a DFL fashion is (Donti et al., 2017); however it assumes the problem is convex thus restricting the applicability of the method. Despite that, PFL approaches can still be applied. While in principle one could train an ML model by minimizing the MSE, we devised a PFL approach that is more suitable in a stochastic optimization setup: we trained a probabilistic model by maximum likelihood estimation. The probabilistic model is a multivariate Gaussian distribution whose means $\mu \in \mathbb{R}^d$ are parameterized by a linear regression model, and whose standard deviations $\sigma \in \mathbb{R}^d$ are a set of trainable but non-contextual parameters, where d is the number of products. In Table 2, we report results for the WSMC with 5 products and 25 items with different penalty values, $\rho = \{1, 5, 10\}$. Similarly to the previous problem, SFGE *achieved lower relative post-hoc regret and infeasibility ratio* and the gap with PFL becomes larger when the problem has a larger penalty coefficient.

We employed the same probabilistic model in conjunction with the Sample Average Approximation (SAA) algorithm (Kleywegt et al., 2002): we rely on the model to gather a set of instance-specific samples, which are subsequently used as scenarios in the SAA algorithm. This approach enables us to obtain a more robust solution. We refer to this pipeline as PFL+SAA. In Figure 1, we show the relative post-hoc regret and the normalized runtime at *inference time* in function of the number of sampled scenarios, for the WSMC previously described and $\rho = 10$. The normalized runtime expresses the relative slowdown in runtime at inference time with respect to SFGE. When increasing the number of scenarios, PFL+SAA improves in relative regret, but at the price of a higher computational cost. As a reference value, we also reported the relative post-hoc regret of SFGE, which does not involve sampling scenarios. Even when 100 scenarios are employed, PFL+SAA *does not outperform* SFGE *and, at the same time, has a computation time at inference time that is at least one order of magnitude larger.*

5. Conclusions

This work aims to widen the applicability of decision-focused learning to tackle predict-then-optimize problems by proposing a method that does not suffer from the limitations of the state of the art. Concretely, we predict *distributions* over problem parameters, which circumvents the zero-gradient problem that occurs when the optimization problem is combinatorial. To estimate the resulting non-zero gradients, we employ SFGE. We showcased the effectiveness of the method on two problems of practical relevance on which no other DFL approach can be applied. In a set of preliminary experiments, SFGE demonstrated superior performance to PFL in terms of both post-hoc regret and infeasibility ratio.

6. Acknowledgements

This work has been supported by the project TUPLES (funded by European Union's Horizon Europe research and innovation programme, GA No. 101070149).

References

Abernethy, J., Lee, C., and Tewari, A. Perturbation techniques in online learning and optimization. *Perturbations, Optimization, and Statistics*, 233, 2016.

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

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

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

Donti, P., Amos, B., and Kolter, J. Z. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017.

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

Elmachtoub, A. N., Liang, J. C. N., and Mcnellis, R. Decision trees for decision-making under the predict-then-optimize framework. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2858–2867. PMLR, 13–18 Jul 2020.

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

Hu, X., Lee, J. C. H., and Lee, J. H. M. Predict+optimize for packing and covering lps with unknown parameters in constraints. *CoRR*, abs/2209.03668, 2022. doi: 10.48550/arXiv.2209.03668. URL https://doi.org/10.48550/arXiv.2209.03668.

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

Mandi, J. and Guns, T. Interior point solving for lp-based prediction+optimisation. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7272–7282. Curran Associates, Inc., 2020.

Mandi, J., Bucarey, V., Mulamba, M., and Guns, T. Decision-focused learning: Through the lens of learning to rank. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14935–14947. PMLR, 17–23 Jul 2022.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. Monte carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62, 2020.

Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Lopez, V. B., and Guns, T. Contrastive losses and solution caching for predict-and-optimize. In *30th International Joint Conference on Artificial Intelligence (IJCAI-21): IJCAI-21*, pp. 2833–2840. International Joint Conferences on Artificial Intelligence, 2021.

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

Paulus, A., Rolínek, M., Musil, V., Amos, B., and Martius, G. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In *International Conference on Machine Learning*, pp. 8443–8453. PMLR, 2021.

Petersen, F. Learning with differentiable algorithms. *arXiv preprint arXiv:2209.00616*, 2022.

Shah, S., Wang, K., Wilder, B., Perrault, A., and Tambe, M. Decision-focused learning without decision-making: Learning locally optimized decision losses. In *NeurIPS*, 2022.

Wilder, B., Dilkina, B., and Tambe, M. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pp. 1658–1665. AAAI Press, 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

A. Additional results

For a more comprehensive analysis, we provide additional results for larger sizes of the KP with uncertain weights and the WSMC. When compared to the sole PFL, we can draw similar conclusions as for the smaller size problems: SFGE always outperforms PFL in terms of both the relative post-hoc regret and infeasibility ratio; moreover, the gap becomes more evident with the increasing of the penalty coefficient value.

Table 3. PFL and SFGE results on the KP with uncertain weights of size 75 for different penalty coefficient values.

Method	*Rel. PRegret*	*Infeas. ratio*
	$\rho = 1$	
PFL	0.115 ± 0.017	0.31 ± 0.05
SFGE	$\mathbf{0.114 \pm 0.014}$	$\mathbf{0.20 \pm 0.04}$
	$\rho = 5$	
PFL	0.247 ± 0.048	0.31 ± 0.05
SFGE	$\mathbf{0.214 \pm 0.058}$	$\mathbf{0.04 \pm 0.02}$
	$\rho = 10$	
PFL	0.440 ± 0.091	0.31 ± 0.05
SFGE	$\mathbf{0.251 \pm 0.064}$	$\mathbf{0.03 \pm 0.02}$

.

Table 4. PFL and SFGE results on the WSMC with 10 items and 50 sets for different penalty coefficient values.

Method	*Rel. PRegret*	*Infeas. ratio*
	$\rho = 1$	
PFL	2.35 ± 0.85	0.98 ± 0.01
SFGE	$\mathbf{1.94 \pm 0.50}$	$\mathbf{0.94 \pm 0.05}$
	$\rho = 5$	
PFL	12.17 ± 4.73	0.98 ± 0.01
SFGE	$\mathbf{4.87 \pm 1.15}$	$\mathbf{0.64 \pm 0.08}$
	$\rho = 10$	
PFL	22.41 ± 7.90	0.98 ± 0.01
SFGE	$\mathbf{7.08 \pm 1.29}$	$\mathbf{0.54 \pm 0.14}$

When compared to PFL+SAA, SFGE probes its efficiency in terms of computational cost at inference time. PFL+SAA provides better relative post-hoc regret but it requires sampling a non-negligible number of scenarios. This is especially visible when $\rho = 10$: even if the post-hoc regret is slightly improved, 100 scenarios are sampled at the price of more than 50 times the computational time of SFGE.

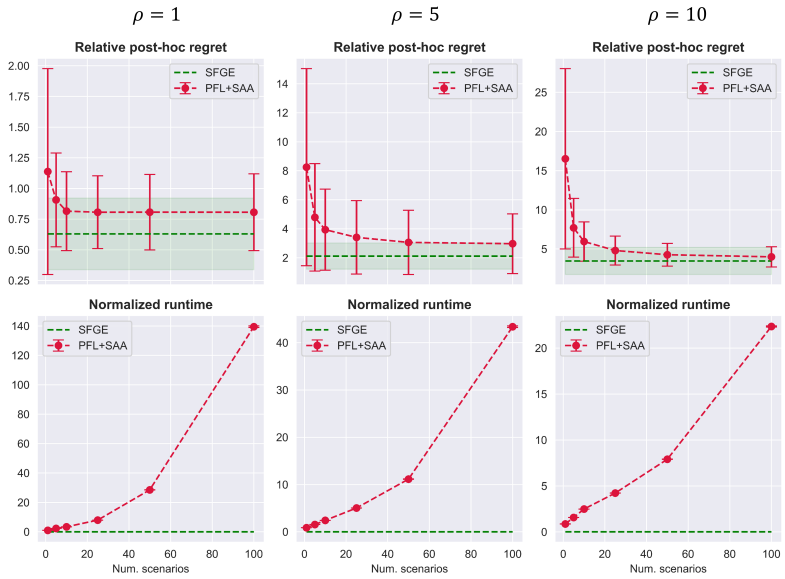

Figure 2. Comparison between SFGE and PFL+SAA on the WSMC of size 5×25.

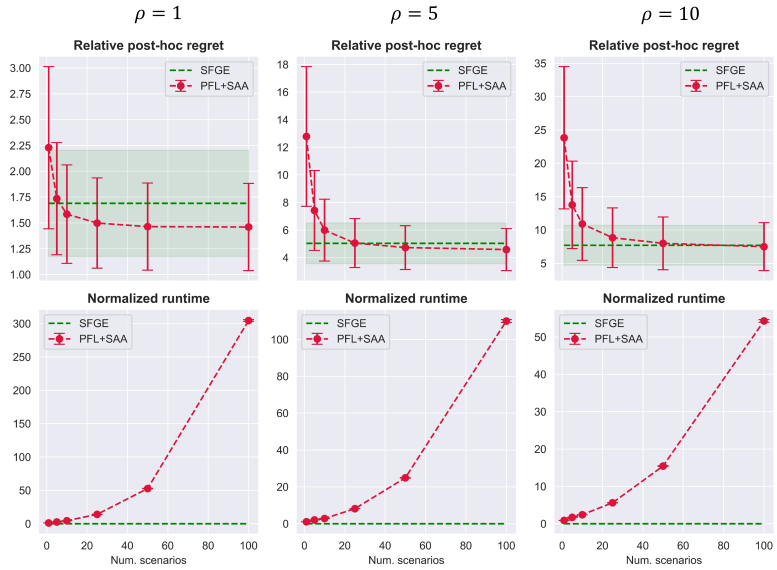

Figure 3. Comparison between SFGE and PFL+SAA on the WSMC of size 10×50.