SCALABLE DECISION FOCUSED LEARNING VIA ONLINE TRAINABLE SURROGATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision support systems often rely on solving complex optimization problems that may require to estimate uncertain parameters beforehand. Recent studies have shown how using traditionally trained estimators for this task can lead to suboptimal solutions. Using the actual decision cost as a loss function (called Decision Focused Learning) can address this issue, but with a severe loss of scalability at training time. To address this issue, we propose an acceleration method based on replacing costly loss function evaluations with an efficient surrogate. Unlike previously defined surrogates, our approach relies on unbiased estimators – reducing the risk of spurious local optima – and can provide information on its local confidence – allowing one to switch to a fallback method when needed. Furthermore, the surrogate is designed for a black-box setting, which enables compensating for simplifications in the optimization model and accounting for recourse actions during cost computation. In our results, the method reduces costly inner solver calls, with a solution quality comparable to other state-of-the-art techniques.

1 INTRODUCTION

Many real-world decision support systems, in domains such as logistics or production planning, rely on the solution of constrained optimization problems; frequently, these problems feature parameters that are estimated based on contextual information via Machine Learning (ML) predictors, an approach sometimes referred to as Prediction Focused Learning (PFL). In past years it has been shown how this common practice can lead to poor decision quality, due to a misalignment between the training objective (usually likelihood maximization) and the actual decision cost. Such an observation has lead to Decision Focused Learning (DFL) (Amos & Kolter, 2017; Elmachtoub & Grigas, 2022), which attempts to correct for this issue by training predictors for minimal decision regret.

While remarkable progress in the field has been made (Mandi et al., 2024), we argue that three issues still prevent DFL methods from finding widespread practical application. First, their *training scalability is often severely limited*, since the problems encountered in decision support are frequently difficult (NP-hard or worse) and most DFL approaches require frequent solver calls and cost evaluations. Second, many DFL methods make somewhat *restrictive assumptions* on the decision problem (e.g. linear cost function, no parameters in the constraints); theoretical analysis and recent experiments (Hu et al., 2022; Elmachtoub et al., 2023) also show that, when the parameter expectations can be accurately estimated, such assumptions erode most of the advantage over PFL. Third, several DFL methods require *explicit knowledge of the problem structure or the solver state*, which in a practical setting would require costly refactoring of the existing tools, or even a radical change of the solution technology. Solving these issues would allow one to use DFL for improving the effectiveness and robustness of *any real-world decision support tool*, while maintaining scalability.

In this work, we aim at making a significant step toward addressing these limitations, by relying on a well-crafted, efficient, surrogate to replace most solver calls at training time. We design our surrogate to be an (asymptotically) unbiased estimator of the regret function, and employ stochastic smoothing and importance sampling to address the 0-gradient problem often occurring in DFL settings. We also include uncertainty quantification by relying on probabilistic models that provide a confidence level, so as to determine when the surrogate should be used or updated and hence balance model exploration and surrogate exploitation at training time. Furthermore, our surrogate is suitable for a black-box setting, where no restrictive assumption is made on how the regret computation is performed and access to the solver state is not needed.

Only a few of the existing DFL methods can be applied to achieve similar goals. A representative set of these is used as a baseline in our empirical evaluation, where we compare DFL and PFL on extended versions of standard benchmarks in the current literature. Specifically, our setups allows for scaling the problem complexity, so as to assess how the evaluated approaches behave on problems of different size (in terms of number of variables or parameters). In our results, our surrogate significantly reduces both the training runtime and the number of solver calls and cost evaluations. We also show that this acceleration, unlike previous attempts in the literature, does not adversely affect the decision quality, which remains comparable to other state-of-the-art techniques.

2 Related Work

In the context of DFL problems where parameters are predicted by a machine learning model, initial works focused on implicit differentiation of the KKT conditions for optimality. In particular, Amos & Kolter (2017) handled convex quadratic programming, while Agrawal et al. (2019) extended it to conic programs. However, these initial methods were unsuitable for combinatorial problems, characterized by piecewise constant loss functions and uninformative gradients. Subsequent studies such as Wilder et al. (2019); Mandi & Guns (2020) addressed MILP problems, by proposing to smooth the loss function through a regularization term (respectively L2 and log-barrier) computed over the decision variables. Other approaches introduced surrogate losses to overcome the zero-gradient problem. Elmachtoub & Grigas (2022) formalized the SPO+ loss as a regret upper bound, Mulamba et al. (2020) proposed the noise-contrastive estimation, Mandi et al. (2022) turned the problem into learning to rank on a pool of solutions. Many of these works also adopted a LP relaxation to speed up the training, which can however adversely affect the final solution quality, and it requires formulating the decision problem as a MILP. A surrogate loss, based on directional gradients, is also proposed in Huang & Gupta (2024) and proved to provide unbiased gradient estimates, which allows it to outperform earlier approaches for strongly misspecified ML models. Unlike our approach, however, this method is restricted to problems with linear cost functions and is actually more computationally expensive at training time, due to the use of a zeroth-order gradient approximation. Finally, the approaches mentioned so far, with the exception of the KKT-based solutions, do not allow for predicted parameters appearing in the problem constraints. Attempts to cover the latter case include Paulus et al. (2021); Hu et al. (2023b;c;a), which require access to the problem formulation and either dedicated solvers or access to the solver internal state.

Only a few DFL approaches target the setting considered in this paper, where no assumption is made on the problem structure and training time scalability is emphasized; this is typically done by replacing the decision cost loss with fast-to-evaluate, differentiable, and learnable surrogates. The first studies in this class include Chung et al. (2022); Lawless & Zhou (2022), which used simple loss approximations with poor results. Recent advances are represented by the convex learnable surrogates proposed by Shah et al. (2022; 2024), and LANCER (Zharmagambetov et al., 2023), which employs a trainable neural network surrogate, fine-tuned at training time similarly to actor-critic approaches in reinforcement learning. These methods are closest to the one we propose, and differ mainly for their use of biased estimators and the lack of local confidence estimation.

DFL problems can also be formulated by treating the decision problem similarly to a reinforcement learning environment, as suggested by Silvestri et al. (2022). A notable example is the Score Function Gradient Estimation (SFGE) method by Silvestri et al. (2023), inspired by Donti et al. (2017); Pogančić et al. (2019); Berthet et al. (2020); Mohamed et al. (2020) and Niepert et al. (2021), which combines stochastic smoothing and policy gradient methods. While we rely on Gaussian processes like Char et al. (2019), we use separate models and exploit contextual information via sample sharing, which simplifies the learning process and avoids length-scale and kernel issues.

3 Problem Formulation

We consider a generalization of a DFL setting, where the parameters y of an optimization problem (e.g. demands) are not known at decision time, but can be estimated based on contextual information x (e.g. hour of the day). Formally, let X and Y be random variables with support D_x and D_y, representing respectively the contextual information and the uncertain parameters, and correlated according to their joint distribution $P(X, Y)$. At decision-making time, the problem parameters are estimated via a predictive model $h_\theta : D_x \rightarrow D_y$, with parameter vector θ. Based on the estimator output $\hat{y}$, we compute a decision vector z^* by solving a constrained optimization problem:

$$z^*(\hat{y}) = \arg\min_z \{f(\hat{y}, z) \mid z \in C(\hat{y})\} \tag{1}$$

where $f : D_y \times D_z \rightarrow \mathbb{R}$ is the problem cost function and $C : D_y \rightarrow 2^{D_z}$ is a constraint function that denotes the feasible space. We treat z^* as a function, assuming that a tie-breaking rule is used when multiple optimal solutions exist. Once the decisions are executed, their quality is determined by means of a second "true" cost function $g : D_y \times D_z \rightarrow \mathbb{R}$. Specifically, $g(y, z)$ represents the cost incurred by the solution z, under a realization y sampled from $P(Y \mid x)$.

The use of a distinct function g for decision quality evaluation distinguishes our setup from those typically used in DFL, and enables compensating for *misspecified decision problem models* – as opposed to misspecified predictors as in Huang & Gupta (2024). These can stem from treating uncertain parameters (e.g. travel times or demands) as deterministic, from disregarded minor constraints, or from approximated non-linearities – all common techniques to ensure scalability in real-world applications. This choice also allows us to deal with estimated parameters in the problem constraints in eq. (1), assuming that infeasible solutions can be repaired at an additional cost.

We wish to train the predictive model for minimal decision regret:

$$\arg\min_\theta \mathrm{E}_{x,y \sim P(X,Y)} [r(y, \hat{y})] \tag{2}$$

where $\hat{y} = h_\theta(x)$ and $r(y, \hat{y}) = g(y, z^*(\hat{y})) - g(y, z^*(y))$. In practice, the expectation is approximated by an average over a training sample $\{x_i, y_i\}_{i=1}^m$, thus leading to:

$$\arg\min_\theta \frac{1}{m} \sum_{i=1}^m r(y_i, \hat{y}_i) \tag{3}$$

where $\hat{y}_i = h_\theta(x_i)$. Solving eq. (3) via first-order methods, as typically done with neural networks predictors, requires computing the gradient of the regret function, which is given by:

$$\frac{\partial}{\partial \theta} r(y_i, \hat{y}_i) = \frac{\partial g}{\partial z^*} \frac{\partial z^*}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \theta} \tag{4}$$

When the optimization problem is linear or combinatorial, or when the g function is piecewise-constant, the gradient might be undefined or null on a large part of the predicted parameter space. Furthermore, evaluating the regret function requires computing z^* and g once per example and per training epoch, which can be prohibitively expensive, when the optimization problem is NP-hard, or the true cost function is based on optimization or simulation.

4 Methodology

We now introduce our method, whose main goal is accelerating the training problem of eq. (3). Similarly to Shah et al. (2022; 2024), we reduce the runtime by replacing, for every training example, the computationally heavy loss function $r(y_i, \hat{y})$ with a faster, trainable, surrogate loss $\tilde{r}_i(\hat{y})$. Formally, for a given realization y_i, associated to the i-th training example, the surrogate model is a function $\tilde{r}_i : D_y \rightarrow \mathbb{R}$ mapping a prediction vector $\hat{y}$ into a corresponding loss value.

We identify three desirable properties for such a function. First, the surrogate should be *differentiable and have informative gradients everywhere*, to support gradient-descent optimization. Second, $\tilde{r}_i$ should be capable of providing *unbiased gradient estimation* to ensure that, if enough calibration data is available, the local optima of the regret loss are preserved. The surrogate losses from Chung et al. (2022); Lawless & Zhou (2022); Shah et al. (2022; 2024) do not satisfy this criterion, running the risk of getting trapped in spurious local minima. Third, $\tilde{r}_i$ should provide *confidence information* for online refinement; namely, one should be able to determine when the surrogate is reliable, and when instead a fallback method based on direct evaluation of z^* and g should be employed.

We propose using Gaussian Processes (GP) with Radial Basis Function (RBF) kernels as the surrogate model, since they satisfy almost all the desired properties. In particular, GPs with RBFs are fully differentiable and support efficient evaluation. Moreover, GPs are (asymptotically) unbiased estimators: given enough samples, they can approximate any function with arbitrarily high precision – though they are biased towards zero-centered predictions with scarce data. Finally, GPs have a distributional output and naturally provide confidence information.

Stochastic smoothing The only property that GPs lack is tied to their nature as unbiased estimators. On the one hand, their are capable of accurately approximating the regret function; on the other, they risk inheriting some of its undesirable traits, such as 0 (or near-0) gradients on large swathes of the input space. One possible solution to this issue is to apply stochastic smoothing to the loss function – somewhat similarly to Silvestri et al. (2023). Specifically, our surrogate approximates a *smoothed* version of r_i, here called $\bar{r}_i$, referred to as $\bar{r}_i : D_y \to \mathbb{R}$ and defined as:

$$\bar{r}_i(\hat{y}) = \mathrm{E}_{\hat{y}' \sim \mathcal{N}(\hat{y}, \sigma)} \left[r_i(\hat{y}') \right] \tag{5}$$

The original loss value is replaced with its expectation under random, Normally distributed, perturbations of its input. The standard deviation σ is a controllable parameter in the method and represents the degree of smoothing. Small values of σ result in a more accurate regret approximation – at the cost of possibly weaker gradients – while higher σ values can provide more informative gradients – but may result in altered local optima. The effect of smoothing is depicted in fig. 1.

In principle, $\bar{r}_i$ can be computed via a simple Monte Carlo approach. In practice, this is highly inefficient, because many samples would be needed for each input $\hat{y}$, and evaluating $r_i(\hat{y}')$ for each sample requires computing both z^* and g. We overcome this limitation by relying on importance sampling to perform multiple computations of $\bar{r}_i(\hat{y})$ based on the same set of observations. Formally, let $\{\hat{y}'_k\}_{k=1}^n$ be a set of predictions for which the value of $r_i(\hat{y}'_k)$ is known. We assume each of them has been sampled according to a distinct process, and refer to $\{q_k\}_{k=1}^n$ as the corresponding probabilities. These samples can be associated to an aggregated probability density function q, defined as a kernel density estimator, and used to define the importance weight function $w(\hat{y}, \hat{y}')$:

$$q(\hat{y}) = \frac{1}{n} \sum_{k=1}^{n} q_k(\hat{y}) \qquad\qquad w(\hat{y}, \hat{y}') = \frac{\phi(\hat{y}'; \hat{y}, \sigma)}{q(\hat{y}')} \tag{6}$$

where $\phi(\cdot; \hat{y}, \sigma)$ is the density for a Normal distribution centered on $\hat{y}$ and having standard deviation σ. Then we have:

$$\bar{r}_i(\hat{y}) \approx \sum_{k=1}^{n} \frac{w(\hat{y}, \hat{y}'_k)}{\sum_{h=1}^{n} w(\hat{y}, \hat{y}'_h)} r_i(\hat{y}'_k) \tag{7}$$

In practice, whenever a new prediction $\hat{y}'_k$ is evaluated during the training process, we store both its regret value and the probability according to which it was sampled; then we estimate the *smoothed* regret associated to the same predictions via eq. (7); finally, we use the smoothed values as targets when training our GPs. This approach allows to increase the sampling efficiency and decrease the variance associated to the computation of $\bar{r}_i$; at the same time it adds the flexibility to control the smoothing level by manipulating the sampling distributions. It is worth noting that, when a limited number of sampling points is available, the natural smoothing from the GPs combines with stochastic smoothing, leading to very regular landscapes for the surrogate loss. In fig. 1 we show an example of stochastic smoothing via importance sampling computed on a set of one-dimensional points sampled from Normal distributions.

Sample sharing Using a distinct surrogate for each training example permits unbiased regret estimation, at the same time limiting the number of input dimensions for the GPs. However, this also prevents information sharing among surrogates. With the aim to further reduce the number of r_i function evaluations, we devised an optional technique to enable sample sharing between different surrogates, if the corresponding regret landscapes are similar. Specifically, at the beginning of the training process, we perform Latin Hypercube Sampling (LHS) to collect a number points $\{\hat{y}_k\}_{k=1}^m$ in the prediction space D_y. We then associate each training example with a vector v containing the regret value computed for each such collected point, i.e. $v_i = \{r_i(\hat{y}_k)\}_{k=1}^m$. It can be seen that two samples i and j are associated to the same regret landscape iff:

$$\lim_{m \to \infty} \|v_i - v_j\|_2 = 0 \tag{8}$$

i.e., if the Euclidean distance between the corresponding vectors converges to 0, as the number of sampled points grows. Accordingly, we measure the similarity between the regret landscapes for

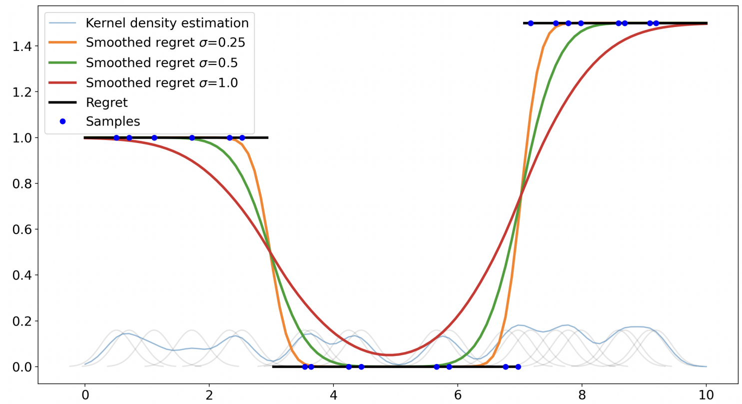

Figure 1: Illustration of stochastic smoothing with varying σ: larger values cause smoother functions with respect to the original regret loss. Smoothed functions are computed using importance sampling over a set of points sampled from Normal distributions.

two samples i and j in terms of such distance. It is worth noting that measuring distances on the contextual information space alone, as done by Shah et al. (2024), introduces a bias in the surrogates, since in a stochastic setting the same x input might be associated with different realizations and consequently with different loss landscapes.

We use the discussed similarity score to determine how relevant the data from one sample is for another sample. This is achieved by operating on the covariance matrix of each GP, by downscaling the kernel outputs. Specifically, let $\hat{y}'_i$ and $\hat{y}''_j$ be two predictions collected respectively for the training examples i and j; then we have:

$$K_{new}(\hat{y}'_i, \hat{y}''_i) = \frac{K_{old}(\hat{y}'_i, \hat{y}''_j)}{1 + e^{\alpha} \|v_i - v_j\|_2} \tag{9}$$

where α is a learnable scale factor and the kernel value is unaltered if $i = j$, since the corresponding distance is 0. The collection phase required for this sample sharing process is also useful to initialize the set of GPs, and to perform standardization of the regret values that are used for their training.

Algorithm We now dive into the concrete implementation of the main training procedure (see algorithm 1), starting from the outer training loop to solve eq. (3), and then moving to the inner training to optimize the GP surrogates.

Our surrogate loss should be combined with a fallback method, ideally one suitable for a general setting where no restrictive assumption on the regret function is made. One such example is the Score Function Gradient Estimation (SFGE) approach by Silvestri et al. (2023). Using this method has two additional benefits. First, SFGE also relies on stochastic smoothing, so that the set of predictions used for training the GPs can be naturally populated, their distribution of origin is known when computing $q(\hat{y})$, and the loss is semantically consistent between the surrogate and the fallback method. Second, the compound smoothing achieved by our GPs when few samples have been collected tends to compensate for the high variance and slow convergence of SFGE. That said, it should be possible to use our surrogate loss with a different fallback method, such as those from Hu et al. (2023a); Elmachtoub & Grigas (2022) or Huang & Gupta (2024).

The first step in algorithm 1 is a pretraining stage where we initialize the GPs and we collect pairs $(\hat{y}, r_i(\hat{y}))$ via LHS. We scale the number of points to be sampled logarithmically with respect to the dimensionality of D_y. We use these points to compute statistics to normalize the input $\hat{y}$ (as GPs expect 0-mean input values) and to standardize the output $r_i(\hat{y})$. Then, for each training example, we employ the associated GP to compute a predicted mean and standard deviation. If the latter is below a threshold β (i.e., the GP is confident with respect to β), we take the mean as a surrogate loss and differentiate through the GP; otherwise, we call the SFGE procedure and we add the generated sample (and its generating distribution) to the corresponding GP. We aggregate loss terms for each training instance into a single loss, mixing values from the surrogates and the fallback method, before gradient computation.

Algorithm 1 Training loop – gradient computation

```
gp ← initializeGPs(y)
for epoch in EPOCHS do
    loss ← 0
    i ← 0
    while i < m do
        ŷ_i ← h_θ(x_i)
        r̄_i, σ_i ← gp_i(ŷ_i)
        if σ_i < β then
            loss ← loss + r̄_i
        else
            q_i ← N(ŷ_i, σ)
            ŷ_i ∼ q_i
            gp_i.add(ŷ_i, r_i(ŷ_i), q_i)
            loss ← loss + SFGE(ŷ_i, r_i(ŷ_i), q_i)
        i ← i + 1
    gradient ← loss.backward()
```

We trigger surrogate training when at least t new sample-target pairs are available. New data points are then shared between similar GPs, if such option is enabled, according to a maximum tolerable Euclidean distance d_{max}. We replace the $r_i(\hat{y})$ targets with the smoothed ones $\bar{r}_i(\hat{y})$, using eq. (7) on the collected samples and distributions. We also apply data preprocessing by normalizing the inputs and standardizing the outputs, on the basis of the statistical information extracted in the pretraining phase. We train GPs by maximum likelihood estimation, with a classical RBF kernel and a length-scale prior $l_i \sim LogNormal(\log(\dim(Y))/2, 1)$. We add this last regularization term, following Hvarfner et al. (2024), to improve scalability in higher dimensions. Finally, the RBF kernels are warm-started in all subsequent trainings, so as to speed up their training. The code for our method is publicly available at `currently-in-the-supplemental-material`.

5 Experiments

We now discuss the experimental analysis conducted to assess the robustness, reliability, accuracy, and especially scalability of our method. We designed experiments to answer four main research questions. **Q1.** How does our surrogate loss perform in terms of decision quality compared to the relevant baselines? **Q2.** How many calls to the black-box solver does it require? **Q3.** How much runtime does it take to converge compared to the other methods? **Q4.** Can it be scaled to high dimensions?

Benchmarks We consider three problem classes, many of which include recourse actions to repair violated constraints, thus leading to misspecified, realistic, decision problems.

Knapsack (KP). We generate 1-0 KP datasets following the procedure proposed by Elmachtoub & Grigas (2022) to model a stochastic mapping between input features x and ground-truth targets y, with a polynomial degree $deg = 5$, number of input features $dim(X) = 5$, a noise half-width $\bar{\epsilon} = 0.5$, and a Poisson distribution to correlate x and y . In our experiments we build datasets on a target space of $dim(Y) = 50$ items. We make use of three different setups, respectively injecting uncertainty (i.e., stochastic correlation) into weights, values and capacity. We also adopt the recourse action system by Silvestri et al. (2023), with a fixed penalty $p = 10$.

Weighted Set Multi-Cover (WSMC). We create WSMC datasets with $dim(X) = 5$ input features, $dim(Y) = 10$ items and $s = 50$ sets, following the guidelines by Grossman & Wool (1997) to generate realistic availability matrices and the same stochastic correlation as in Elmachtoub & Grigas (2022). We use the recourse action adopted by Silvestri et al. (2023), with a fixed penalty $p = 10$.

Toy. We define a synthetic toy dataset. In this setting the input features are deterministically mapped to a target space, using a weight matrix $W \in R^{dim(Y) \times dim(X)} \sim U(0,1)$, such that $y = Wx$. The underlining optimization problem is a trivial map where $z^*(\hat{y}) = \hat{y}$, while the cost is given by a pseudoconvex piecewise step function with minimum in y: $g(y, z^*(\hat{y})) = s\lfloor \|y-\hat{y}\|_2/l \rfloor$, where s

controls the step heights and l determines the distance between steps. We set $s = 5$ and $l = 1$ in our experiments. This new benchmark provides a controlled way to investigate convergence properties, as the entire loss landscape is known. Additionally, it allows for changing the dimensionality of Y, while keeping a negligible cost for computing z^*, so to stress DFL techniques on larger scales.

Baselines Our baselines include a predictive model trained for maximum likelihood, referred to as Prediction Focused Learning (PFL), plus state-of-the-art DFL methods that apply to black-box settings. In particular, we include in our comparison the SFGE method by Silvestri et al. (2023), which also serves as fallback for our approach, allowing us to directly assess the impact of the surrogate-based acceleration. For this method, we use a trainable parameter for σ, with starting value 0.1, and we warm start the predictor via PFL training. We then consider EGL, LODL, and LANCER, respectively from Shah et al. (2024), Shah et al. (2022) and Zharmagambetov et al. (2023), as representative of other state-of-the-art black-box surrogate-based approaches. We employ the same set of hyperparameters proposed by the authors, implementing all the four convex surrogates (MSE, Directed-MSE, Quadratic, Directed-Quadratic) and fixing the number of samples to 250 for LODL and to 32 or 42 for EGL, to put it on par with our model in terms of calls, for a fair regret comparison. We adopt the same surrogate model architecture proposed in Zharmagambetov et al. (2023) for LANCER, with two hidden layers of 200 units and tanh activation functions; we set $t = 10$ for the dual training. As a representative of a state-of-the-art DFL method requiring restrictive assumptions, we include SPO+ by Elmachtoub & Grigas (2022), where applicable.

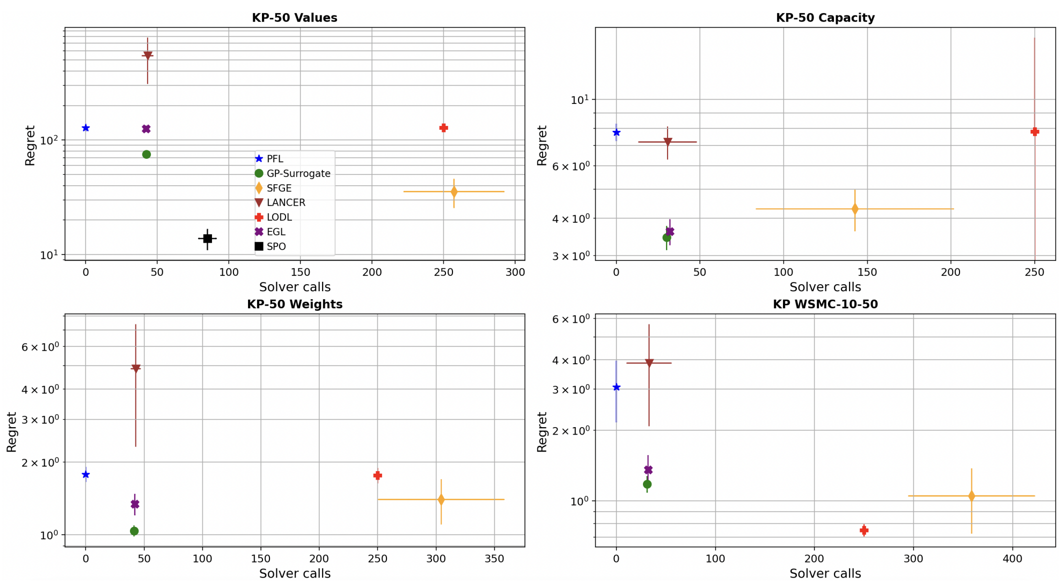

Figure 2: Regret and solver calls across the benchmarks: points represent the average regret (on a logarithmic scale) and average solver calls per instance for all the baselines. Lines represent standard deviations for each method. Only the best (lowest regret) models are reported for LODL and EGL.

Results In this section we report results on a set of experiments designed to answer the discussed research questions. For each setting we generate 5 datasets, each one containing 1000 instances, differently split into train (80%), validation (10%) and test sets (10%). All the models are trained with the Adam algorithm by Kingma (2014) and a learning rate $lr = 10^{-3}$, using an early stopping criterion on regrets computed on the validation set, to avoid overfitting. We enable stochastic smoothing, differentiation, and pretraining for our method, here named GP-Surrogate. By default, we keep sample sharing off, as it is best suited to improve speed on very time-consuming problems, at the cost of solution quality (see table 3 for more details). We set $\beta = 1.0$ as a tradeoff between speed and precision in learning, as shown in appendix C, and $t = 40$. All the experiments were run on an Apple M3 Pro CPU with 12 cores and 18GB of RAM.

Table 1: Runtime and average regret for GP-Surrogate and SFGE on WSMC-10 with 250, 500, 750 and 1000 sets. Time limit is $900s$.

	Method	*WSMC-10-250*	*WSMC-10-500*	*WSMC-10-750*	*WSMC-10-1000*
Runtime	GP-Surrogate	$\mathbf{289.24 \pm 6.54}$	$\mathbf{459.70 \pm 15.73}$	$\mathbf{638.50 \pm 14.70}$	$\mathbf{818.44 \pm 17.74}$
	SFGE	$900.0^* \pm 0.0$	$900.0^* \pm 0.0$	$900.0^* \pm 0.0$	$900.0^* \pm 0.0$
Regret	GP-Surrogate	$\mathbf{1.55 \pm 0.57}$	$\mathbf{1.51 \pm 0.50}$	$\mathbf{1.48 \pm 0.54}$	$\mathbf{1.51 \pm 0.59}$
	SFGE	1.83 ± 0.86	2.44 ± 1.05	2.82 ± 1.20	3.54 ± 1.25

Q1. We compare the regret for each approach on the KP with 50 items (KP-50) and uncertainty in weights, values and capacity, and on the the WSMC with 10 items and 50 sets (WSMC-10-50). Results, summarized in fig. 2 and presented in extended form in appendix A, indicate that using our surrogate on top of SFGE leads to solutions of similar quality (or even better) in all but one benchmark, and with less variability. GP-Surrogate provides better regret and stability than the best approaches in the LODL and EGL class. We also found the performance of LODL and EGL to be inconsistent w.r.t. the employed convex surrogate, as highlighted in appendix A. Overall, our method seems to provide a better alternative for accelerating training than these approaches, at least in terms of regret, most likely thanks to its lack of a structural estimation bias, and to the ability to switch to the fallback method in case of low confidence. The surrogate model used by LANCER seems often unable to reasonably approximate the real loss landscape, causing inconsistent training results and high regrets (even more than PFL), despite a significant reduction in terms of solver calls. The SPO method, when applicable, significantly outperforms all the others, which suggests that traditional DFL approaches should still be preferred to black-box ones when permitted by their assumptions.

Q2. We adopt the same datasets as in Q1 to analyze the computational cost of all approaches. Results are again depicted in fig. 2. In appendix B we report the number of solver calls (per training sample), and the total runtime for each method. The former metric should be considered more important, as for any sufficiently difficult problem the decision time will be the dominant factor. GP-Surrogate reduces the number of calls by almost an order of magnitude compared to its fallback method, greatly increasing viability in a practical setting. We align the number of EGL samples to the solver calls from our method and observe EGL performing significantly worse in terms of regret. The LODL approaches, in their reference configuration, perform more solver calls for even worse regret in all but one benchmark. The same considerations apply for the runtime.

Q3. We evaluate scalability for complex problems with time-consuming solver calls, by building versions of the WSMC-10 datasets with an increasing number of item sets. For this experiment we set a time limit of $900s$ per training attempt, to simulate real-world scenarios with limited computational resources. We compare GP-Surrogate with SFGE and report the results in table 1. As it can be seen, GP-Surrogate consistently achieves training convergence. SFGE fails to do so even in the simplest cases, which causes the approach to have worse regret than GP-Surrogate in this case.

Q4. We evaluate scalability for high-dimensional predictions (as opposed to larger-size problems like in Q3), by increasing the number of decision variables in the Toy benchmark. We report results in table 2. SFGE gets stuck into local optima, likely because of the nature of this synthetic problem, which requires strong variations of σ across the training steps. Conversely, GP-Surrogate is still able to effectively minimize the loss function. This behavior shows that our surrogates can enable convergence to high-quality solutions, even in high-dimensional spaces. We conjecture this is partly due to the fact that, since the realization y is always used when training the GPs, they can naturally identify the presence of a local minimum for r_i at that location; this property is shared with the surrogates from Shah et al. (2022), but with all the discussed benefits of our solution.

Overall, the experiments indicate that our approach can outperform its fallback method in terms of solver calls and total runtime, thus making it scalable to complex problems, while keeping a comparable and sometimes better decision quality. The degree of acceleration is similar to the LODL, EGL, and LANCER models, but with improved consistency, reliability, and solution quality. Finally, the method scales much better then SFGE on high-dimensional parameter spaces, thanks to the combination of stochastic and GP smoothing.

Table 2: Average regret for GP-Surrogate and SFGE on the Toy dataset with 64, 128, 256 and 512 dimensions.

	Method	*Toy-64*	*Toy-128*	*Toy-256*	*Toy-512*
Regret	GP-Surrogate	$\mathbf{5.61 \pm 2.97}$	$\mathbf{4.18 \pm 0.96}$	$\mathbf{1.29 \pm 0.46}$	$\mathbf{0.29 \pm 0.23}$
	SFGE	39.78 ± 3.06	58.61 ± 1.02	113.28 ± 3.41	271.82 ± 9.49

Table 3: Average regret and average calls per instance for GP-Surrogate with different components on KP-50 with uncertain weights, values and capacity and WSMC-10 with 50 sets.

	Method	*KP-50 weights*	*KP-50 values*	*KP-50 capacity*	*WSMC-10-50*
Regret	FULL MODEL	1.14 ± 0.11	106.38 ± 10.19	3.52 ± 0.33	1.39 ± 0.23
	SMOOTHING OFF	1.40 ± 0.11	149.67 ± 16.68	3.64 ± 0.52	1.42 ± 0.19
	PRETRAIN OFF	1.15 ± 0.08	$\mathbf{57.83 \pm 2.61}$	5.04 ± 0.33	$\mathbf{0.67 \pm 0.10}$
	SAMPLE SHARING OFF	$\mathbf{1.04 \pm 0.05}$	75.07 ± 5.20	$\mathbf{3.45 \pm 0.31}$	1.18 ± 0.09
	DIFFERENTIATION OFF	1.63 ± 0.21	156.56 ± 25.23	3.61 ± 0.57	1.58 ± 0.71
Slv. calls	FULL MODEL	$\mathbf{39.84 \pm 2.51}$	$\mathbf{35.41 \pm 0.48}$	$\mathbf{6.93 \pm 0.14}$	$\mathbf{28.31 \pm 1.10}$
	SMOOTHING OFF	41.58 ± 0.74	36.44 ± 0.35	6.97 ± 0.12	29.20 ± 0.59
	PRETRAIN OFF	121.71 ± 3.71	88.34 ± 2.55	48.91 ± 0.60	72.65 ± 9.41
	SAMPLE SHARING OFF	41.33 ± 1.22	42.32 ± 0.09	30.04 ± 0.05	31.06 ± 0.35
	DIFFERENTIATION OFF	42.40 ± 4.69	36.28 ± 0.48	6.95 ± 0.15	28.95 ± 1.03

Ablation studies To prove the effectiveness of all the major components of our method, we also conducted an ablation study by separately disabling stochastic smoothing, pretraining, sample sharing and GP differentiation. Results, reported in table 3, reveal that in all the settings removing differentiation or smoothing causes higher average regrets and solver calls, highlighting their relevance. Results with no sample sharing are the same of fig. 2; we note that sharing points between GP models affects negatively the average regrets, but reduces the average calls. In most cases, this extra source of acceleration is not enough to justify the lower decision quality, but it can be extremely valuable in some cases, as observed in the KP-50 capacity benchmark where the number of solver call is almost two orders of magnitude lower than SFGE. For what concerns pretraining, the number of solver calls grows sensibly when it is turned off, proving its fundamental role for learning confident surrogates in early stages. However, relying more on the fallback method does not imply a downgrade in terms of regrets; in fact, in two settings out of four, we see a strong improvement when the surrogate usage is more moderate.

6 CONCLUSIONS

We present an approach to improve the applicability of DFL methods, targeting scenarios where solution and cost computation are time-consuming and where access to the problem structure and solver state is impossible or inconvenient. In this setting, the existing DFL approaches are too slow to converge, they may not be applicable, or they accelerate the training process at the cost of a worse decision quality. We employ a GP-based surrogate loss function, trained online in alternation with a fallback method, to exploit the surrogate speed without loosing the ability to adapt to the regret function landscape. We solve the 0-gradient issue relying on stochastic smoothing via importance sampling – which also motivates our choice of SFGE for the fallback method. Our experimental evaluation reveals that our surrogate matches or outperforms SFGE, while reducing the number of solver calls by up to two orders of magnitude, depending on the problem and the model configuration. Our method improves over related approaches in terms of sample efficiency, solution quality, or both. Finally, we show that the approach can be scaled to higher dimensions. Some potential directions of improvement remain unexplored. A point of particular interest is the possibility to adjust the smoothing factor, or the points where smoothed regret is computed, at training time – without the need to collect additional samples. Moreover, by adapting acquisition functions from classical Bayesian optimization, it might be possible to remove the need for a fallback method. Finally, similar to the LODL, EGL, and LANCER approaches, our learned surrogates could be exported for the construction of new training sets or generally for approximating regret evaluation.

Reproducibility Checklist

Based on the AAAI format:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced: **yes**
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results: **yes**
- Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper: **yes**
- A motivation is given for why the experiments are conducted on the selected datasets: **yes**
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available: **yes**
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting: **partial**
- Any code required for pre-processing data is included in the appendix: **yes**
- All source code required for conducting and analyzing the experiments is included in a code appendix: **yes**
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes: **yes**
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from: **partial**
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results: **yes**
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks: **partial**
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics: **yes**
- This paper states the number of algorithm runs used to compute each reported result: **yes**
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information: **yes**
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank): **no**
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments: **yes**

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.

Ian Char, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew O Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Tsai-Hsuan Chung, Vahid Rostami, Hamsa Bastani, and Osbert Bastani. Decision-aware learning for optimizing health supply chains. *arXiv preprint arXiv:2211.08507*, 2022.

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

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

Adam N Elmachtoub, Henry Lam, Haofeng Zhang, and Yunfan Zhao. Estimate-then-optimize versus integrated-estimation-optimization versus sample average approximation: a stochastic dominance perspective. *arXiv preprint arXiv:2304.06833*, 2023.

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

Xinyi Hu, Jasper Lee, and Jimmy Lee. Two-stage predict+ optimize for milps with unknown parameters in constraints. *Advances in Neural Information Processing Systems*, 36:14247–14272, 2023a.

Xinyi Hu, Jasper CH Lee, and Jimmy HM Lee. Branch & learn with post-hoc correction for predict+ optimize with unknown parameters in constraints. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 264–280. Springer, 2023b.

Xinyi Hu, Jasper CH Lee, and Jimmy HM Lee. Predict+ optimize for packing and covering lps with unknown parameters in constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 3987–3995, 2023c.

Yichun Hu, Nathan Kallus, and Xiaojie Mao. Fast rates for contextual linear optimization. *Management Science*, 68(6):4236–4245, 2022.

Michael Huang and Vishal Gupta. Decision-focused learning with directional gradients. *Advances in Neural Information Processing Systems*, 37:79194–79220, 2024.

Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions. *arXiv preprint arXiv:2402.02229*, 2024.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Connor Lawless and Angela Zhou. A note on task-aware loss via reweighing prediction loss by decision-regret. *arXiv preprint arXiv:2211.05116*, 2022.

Jayanta Mandi and Tias Guns. Interior point solving for lp-based prediction+ optimisation. *Advances in Neural Information Processing Systems*, 33:7272–7282, 2020.

Jayanta Mandi, Vıctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: Through the lens of learning to rank. In *International conference on machine learning*, pp. 14935–14947. PMLR, 2022.

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

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, and Tias Guns. Contrastive losses and solution caching for predict-and-optimize. *arXiv preprint arXiv:2011.05354*, 2020.

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

Anselm Paulus, Michal Rolínek, Vít Musil, Brandon Amos, and Georg Martius. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In *International Conference on Machine Learning*, pp. 8443–8453. PMLR, 2021.

Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2019.

Sanket Shah, Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. Decision-focused learning without decision-making: Learning locally optimized decision losses. *Advances in Neural Information Processing Systems*, 35:1320–1332, 2022.

Sanket Shah, Bryan Wilder, Andrew Perrault, and Milind Tambe. Leaving the nest: Going beyond local loss functions for predict-then-optimize. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14902–14909, 2024.

Mattia Silvestri, Allegra De Filippo, Michele Lombardi, and Michela Milano. Unify: a unified policy designing framework for solving constrained optimization problems with machine learning. *arXiv preprint arXiv:2210.14030*, 2022.

Mattia Silvestri, Senne Berden, Jayanta Mandi, Ali İrfan Mahmutoğulları, Brandon Amos, Tias Guns, and Michele Lombardi. Score function gradient estimation to widen the applicability of decision-focused learning. *arXiv preprint arXiv:2307.05213*, 2023.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1658–1665, 2019.

Arman Zharmagambetov, Brandon Amos, Aaron Ferber, Taoan Huang, Bistra Dilkina, and Yuandong Tian. Landscape surrogate: Learning decision losses for mathematical optimization under partial information. *Advances in Neural Information Processing Systems*, 36:27332–27350, 2023.

A Extended results

We report in table 4 complete results relative to the average regret score for all the models, including all the LODL and EGL variations.

Table 4: Average regret for all the models on KP-50 with uncertain weights, values and capacity and WSMC-10 with 50 sets.

Method	*KP-50 weights*	*KP-50 values*	*KP-50 capacity*	*WSMC-10-50*
PFL	1.78 ± 0.12	128.55 ± 8.99	7.77 ± 0.50	3.06 ± 0.89
GP-Surrogate	$\mathbf{1.04 \pm 0.05}$	75.07 ± 5.20	$\mathbf{3.45 \pm 0.31}$	1.18 ± 0.09
SFGE	1.40 ± 0.29	35.40 ± 9.77	4.30 ± 0.67	1.05 ± 0.32
LANCER	4.85 ± 2.53	545.14 ± 233.63	7.20 ± 0.89	3.87 ± 1.78
LODL-MSE	1.76 ± 0.12	127.78 ± 9.36	7.80 ± 8.24	3.04 ± 0.88
LODL-Quadratic	2.63 ± 0.34	813.19 ± 108.98	9.34 ± 0.35	5.18 ± 1.14
LODL-Directed-MSE	5.85 ± 0.41	315.08 ± 55.10	17.37 ± 3.97	$\mathbf{0.75 \pm 0.04}$
LODL-Directed-Quadratic	2.06 ± 0.21	190.34 ± 5.84	14.96 ± 0.41	2.92 ± 0.89
EGL-MSE	1.78 ± 0.09	125.48 ± 6.69	6.41 ± 1.02	2.25 ± 0.32
EGL-Quadratic	1.34 ± 0.13	247.49 ± 95.72	6.28 ± 0.33	2.43 ± 0.93
EGL-Directed-MSE	1.75 ± 0.12	129.74 ± 3.76	6.13 ± 0.37	2.32 ± 1.42
EGL-Directed-Quadratic	1.97 ± 0.06	275.98 ± 70.70	3.61 ± 0.35	1.36 ± 0.20
SPO	–	$\mathbf{13.78 \pm 2.82}$	–	–

B Solver calls and runtime comparison

In table 5 we show the average solver calls per instance and the runtime for each baseline model.

Table 5: Average calls per instance and runtime for all the models on KP-50 with uncertain weights, values and capacity and WSMC-10 with 50 sets.

	Method	*KP-50 weights*	*KP-50 values*	*KP-50 capacity*	*WSMC-10-50*
Slv. calls	GP-Surrogate	$\mathbf{41.33 \pm 1.22}$	42.32 ± 0.09	$\mathbf{30.04 \pm 0.05}$	$\mathbf{31.06 \pm 0.35}$
	SFGE	304.40 ± 53.80	257.20 ± 34.97	142.60 ± 58.95	358.60 ± 63.62
	LANCER	42.8 ± 3.96	43.2 ± 3.86	30.60 ± 17.24	33.0 ± 22.35
	LODL (All)	250.0 ± 0.0	250.0 ± 0.0	250.0 ± 0.0	250.0 ± 0.0
	EGL (All)	42.0 ± 0.0	$\mathbf{42.0 \pm 0.0}$	32.0 ± 0.0	32.0 ± 0.0
	SPO	–	85.0 ± 6.09	–	–
Runtime	GP-Surrogate	199.40 ± 6.77	136.01 ± 10.02	274.52 ± 66.50	228.73 ± 20.59
	SFGE	717.28 ± 127.26	774.16 ± 173.62	396.93 ± 160.09	1849.74 ± 344.25
	LANCER	207.31 ± 32.24	221.07 ± 20.81	304.38 ± 136.95	348.68 ± 241.32
	LODL (Best)	365.26 ± 37.76	474.75 ± 35.01	623.06 ± 65.38	1034.88 ± 46.12
	EGL (Best)	$\mathbf{63.01 \pm 4.99}$	$\mathbf{72.31 \pm 7.40}$	$\mathbf{82.27 \pm 3.89}$	$\mathbf{118.99 \pm 2.16}$
	SPO	–	180.51 ± 30.30	–	–

C Sensitivity analysis

In table 6 we analyze how β influences results. As expected, higher values lead to an increase in the surrogate loss exploitation. However, we observe a counter-intuitive behavior in average regrets, which improve even if less confident estimations take the place of real regrets. We believe these results my be determined by spurious local optima introduced by the GP loss approximation.

Table 6: Average regret and average calls per instance for GP-Surrogate with different β values on KP-50 with uncertain weights, values and capacity and WSMC-10 with 50 sets.

	Method	*KP-50 weights*	*KP-50 values*	*KP-50 capacity*	*WSMC-10-50*
Regret	$\beta = 0.01$	1.07 ± 0.14	314.02 ± 263.90	3.76 ± 0.63	5.69 ± 2.17
	$\beta = 0.05$	1.03 ± 0.16	474.31 ± 136.58	3.63 ± 0.32	4.05 ± 1.86
	$\beta = 0.1$	$\mathbf{1.01 \pm 0.15}$	176.30 ± 37.33	3.61 ± 0.33	3.87 ± 1.17
	$\beta = 0.5$	1.06 ± 0.12	$\mathbf{74.85 \pm 4.82}$	3.57 ± 0.35	1.77 ± 0.43
	$\beta = 1.0$	1.04 ± 0.05	75.07 ± 5.20	$\mathbf{3.45 \pm 0.31}$	$\mathbf{1.18 \pm 0.09}$
Slv. calls	$\beta = 0.01$	279.16 ± 12.40	148.95 ± 41.43	81.67 ± 21.51	53.65 ± 7.54
	$\beta = 0.05$	156.53 ± 23.56	57.25 ± 3.63	56.84 ± 2.95	71.68 ± 43.56
	$\beta = 0.1$	116.74 ± 13.90	55.52 ± 2.34	41.35 ± 0.84	47.09 ± 2.71
	$\beta = 0.5$	57.75 ± 2.55	$\mathbf{40.29 \pm 0.30}$	36.52 ± 0.13	39.73 ± 1.74
	$\beta = 1.0$	$\mathbf{41.33 \pm 1.22}$	42.32 ± 0.09	$\mathbf{30.04 \pm 0.05}$	$\mathbf{31.06 \pm 0.35}$