Directed Perturbations for Efficient Learning of Surrogate Losses

T.R. Cargan[1], D. Landa-Silva[1], I. Triguero[2,3]

[1]*Computational Optimisation and Learning (COL) Lab, School of Computer Science*
University of Nottingham, United Kingdom, NG8 1BB
[2]*Department of Computer Science and Artificial Intelligence*, *University of Granada*, Granada, Spain
[3]*Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI)*
{timothy.cargan, dario.landasilva}@nottingham.ac.uk, triguero@decsai.ugr.es

***Abstract*—Decision-Focused Learning (DFL) is a paradigm to learn neural network-based predictive models tailored to a specific optimisation problem. A key challenge for DFL methods lies in the non-differentiable nature of most optimisation problems. Recent solutions use a learned, differentiable, model to act as a surrogate loss. To learn the model, the optimisation problem is solved repeatedly using random perturbations of the predictions to calculate a regret value which can be used as a target to learn the surrogate loss. However, this necessitates numerous runs of the, potentially computationally expensive, optimiser. As such, maximising the useful information from each run of the optimizer is paramount. A sample of purely random perturbations may not yield an effective distribution to learning the surrogate from. We propose using a directed perturbation strategy to generate a set of meaningful perturbations for learning a surrogate loss model. We evaluate our approach on a resource allocation problem and real-world case study focused on a critical energy challenge: optimising solar-plus-battery systems. The results show that directed perturbations learn a stronger surrogate loss with fewer runs of the optimiser, enabling a more efficient DFL.**

***Index Terms*—Decision-Focused Learning, Predict-then-Optimise, Machine Learning, Optimisation, Solar Energy Systems**

I. Introduction

Real-world decision-making is inherently complex, often requiring optimal solutions, z^*, based on information, Y, that is unknown at decision time. Consider, for instance, the problem of optimising solar energy systems, where future solar irradiance and energy demand, Y, are uncertain, yet critical for effective energy management and grid integration [1]. We refer to these types of problems as predict-then-optimise. A common approach to solving them in the real world is to use a two-step method [2, 3]. This method typically employs a neural network (NN) as a predictive model, $M(X) \rightarrow \hat{Y}$, to infer the unknown information $\hat{Y}$ from observable features X. Subsequently, an optimisation process $Z^*(\hat{Y}) \rightarrow \hat{z^*}$ uses these predictions to determine the optimal decision $\hat{z^*}$.

While intuitively appealing, the predictive model learned in the two-step method will likely produce prediction errors that lead to suboptimal decisions, i.e., $\hat{z^*} \neq z^*$. Crucially, in combinatorial optimisation (CO) problems with correlated uncertainties [4], model misspecifications – arising from inductive bias, insufficient features X, or imperfect learning – can result in unbounded decision errors [5]. This highlights the need for methods that explicitly account for the downstream task during the model learning phase.

Decision-focused learning (DFL), also referred to as Smart Predict+Optimise, is an emerging approach to solve predict-then-optimise problems [6, 3]. It is a learning paradigm that attempts to minimise the impact of errors, caused due to model misspecification as shown in [5], on the quality of the induced solution. This is achieved by tuning the predictive model for a specific downstream task. By propagating the error induced by the prediction back into the machine learning model, it can be tailored to the downstream optimisation task, acknowledging the uncertainty in real-world predictions. This can often result in a better task-specific performance than using a classical trained predictive model.

A key challenge in DFL is propagating the information about the generated solution and its associated cost, back through the optimisation process so it can be incorporated into the learning step. While one can handcraft surrogates or relax the optimisation problem to make it differentiable, these approaches are often limited to certain types of optimisation problems. A promising solution is to learn differentiable surrogate loss models (potentially implemented as neural networks) that can incorporate information about the optimisation task and allow for the use of non-differentiable black box optimisers. Shah et al. proposed Locally Optimised Decision Losses (LODL) to learn these surrogates; by randomly perturbing problem instances and calculating the decision quality (DQ), a form of regret, they were able to use supervised learning to model the optimiser with a fully differentiable function that was in turn used as the loss function to train a predictive NN [7].

While LODL is effective, generating sufficient training

This publication is part of the TSI-100927-2023-1 Project, funded by the Recovery, Transformation and Resilience Plan from the European Union Next Generation through the Ministry for Digital Transformation and the Civil Service. This work is also supported by the Knowledge Generation Project PID2023-149128NB-I00.

data for these surrogate models is computationally expensive. Solving the optimisation problem for each perturbation of every instance becomes a major bottleneck. Furthermore, purely random perturbations often lead to an imbalanced dataset of DQ values, predominantly clustered near zero, hindering effective learning. Resampling techniques can mitigate this imbalance but introduce even more computation. To address these challenges, we propose a novel directed perturbation strategy based on evolutionary algorithm (EA)s. Our approach strategically guides the sampling process in the DQ space, aiming to generate a more informative and balanced distribution of samples. By maximising the useful information gained from each optimisation run, our method learns stronger surrogate losses with significantly fewer optimiser calls, leading to more efficient DFL. This enhanced sample efficiency is crucial for scaling DFL to complex, real-world problems.

To test the performance of our method, we take a case study in the renewable energy space. Bergmeir et al. present a competition in which the contestants must produce a forecast of solar power generation and energy demand, which are then used as input for a scheduling problem [8]. This problem directly fits into the class of predict-then-optimise, however, none of the proposed solutions employed DFL. To demonstrate our methods, we apply them to a modified version of the solar problem presented by Bergmeir et al.

The key contributions of this work are:

- We introduce a directed perturbation strategy using EAs to improve the sample efficiency of learning surrogate loss models in DFL, leading to more efficient training.
- We demonstrate the effectiveness of our approach on a real-world solar-plus-battery optimisation problem, showcasing its applicability to complex energy management scenarios.

The rest of this paper is structured as follows: Section II discusses related work in DFL. Section III gives the necessary background on prediction, optimisation and DFL. Section IV details our proposed directed perturbation strategy for efficient surrogate loss learning. Section V presents and analyses the experimental results. Finally, Section VI concludes the paper and discusses future directions.

II. Related Work

Attempting to find an optimal solution to a problem conditioned on parameters unknown at inference time is common in the real world [9] e.g resource allocation with unknown demand [10, 8], routing with unknown travel times [11] etc. In order to find a solution, a prediction of the unknown values must be made from observable features. This process can be explicitly split into a predict and an optimise step.

One approach to tackle these types of problems is to use techniques for optimisation under uncertainty such as robust optimisation [12] or stochastic optimisation [2], finding an optimal solution using a distribution rather than a point values of the unknowns. These methods do not consider the upstream model generating the predictions. It has been shown that tailoring the predictive model to the downstream task can yield better task-specific loss [5, 6].

Treating the predict and optimise process as a single problem, is a more holistic approach that integrates both the predictive model and optimisation task. Recent work by [13] reformulates the two stages of predict-then-optimise into a single process that co-learns both the predictive model and optimiser. This is similar to neural methods for solving combinatorial optimisation problems such as [14, 15]. While these methods can be effective, they completely replace existing predict-then-optimise pipelines.

Conversely, Decision-focused learning (DFL), learns a predictive model tuned to the given optimisation task by using a task-specific loss informed by the optimisation step [16, 17]. Works such as [18] use handcrafted losses to learn decision trees. However, for many learning methods, this requires passing gradient information through the optimisation step that generates the solution and cost. Methods such as [16, 19, 20] make use of differentiable optimisation. Work such as [21] use relaxations to make a surrogate. However, they do not generalise to all types of optimisation problems, especially non-convex problems. An alternative approach is to learn a differentiable surrogate model. Approaches such as [7, 22] directly learn a surrogate model for every instance before training the predictive model. This has two benefits, it works for black box optimises and removes the optimiser from the training loop.

III. Background

The classic two-step approach to predict-then-optimise problems is to split them into: 1) A prediction step, using Machine learning (ML) to predict the unknown values; 2) An optimisation step where the solver is conditioned on the predicted values, to find a solution. The two steps are shown in Figure 1. The steps to generate a solution for an instance are defined as [8]:

a) Prediction: Generate predictions of the unknown values (y_i) using a learned model. The model can be trained using supervised ML to approximate the function $\mathcal{F}(x_i) = y_i$ with a model $M_\theta(x_i) \rightarrow \hat{y_i}$; Given a dataset, $\mathcal{D}$, of N feature and label pairs: $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$. We can learn a θ^* that minimises a loss function $\mathcal{L}(\hat{y_i}, y_i)$. The learning process is given by:

$$\theta^* = \arg\min_{\theta} \ \frac{1}{N} \sum_{i=0}^{N} \mathcal{L}(M_\theta(x_i), y_i) \tag{1}$$

Where θ^* are the optimal parameters for the model given the available data and use of a standard loss function such as mean squared error, $\mathcal{L}(\hat{y_i}, y_i) = (\hat{y_i} - y_i)^2$.

b) Optimisation: Find the optimal solution, $\boldsymbol{z_i^*}$, for a problem instance using the fitness function $f(\boldsymbol{z}; \boldsymbol{y_i})$, which is conditioned on the values y_i; and in the solution space $Z \subset \mathcal{Z}$ $s.t.$ $g(\boldsymbol{z}; \boldsymbol{y_i}) \leq 0 \ \forall \ z \in Z$, where $\mathcal{Z}$ contains all possible solutions, is defined as:

$$\boldsymbol{Z^*}(y_i) = \boldsymbol{z_i^*} = \arg\min_{\boldsymbol{z} \in Z} \ f(\boldsymbol{z}; \boldsymbol{y_i}) \tag{2}$$

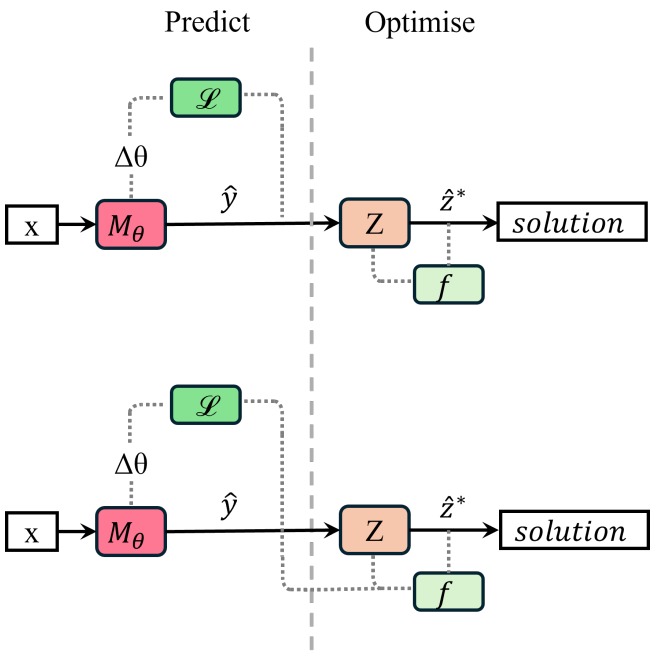


Fig. 1: An overview of the two step method (top) and decision-focused learning (DFL) (bottom). The dark arrows show how predictions flow with the gray dashed lines demonstrating how information is used to update the model.

Computing $\boldsymbol{z}^*$ can be challenging. Calculating the fitness function, f, may be computationally expensive and is often non-convex and may have no closed-form solution, making gradient descent methods impractical. Furthermore, the solution space, Z, may be so large that it is infeasible to try all possible solutions. As such, methods must be used to reduce the search space to find the solution. This is typically a combination of heuristics, defined using domain expertise, and other search methods such as genetic algorithms.

A. DFL

In all relevant predict-then-optimise settings, the true labels, $\boldsymbol{y}$, are unknown at inference time. As such, the optimisation fitness function must be conditioned on $\hat{\boldsymbol{y}}$. The decision induced using the predicted value is $\boldsymbol{Z}^*(\hat{\boldsymbol{y}}) = \hat{\boldsymbol{z}}^*$. In a standard two-step approach (top part of Figure 1), the predictive model learned in the first step has no knowledge about the downstream task, an intermediate loss such as mean square error is used to train the model. While effective at training, the model learns to minimise a prediction error that contains no information about the downstream optimisation task. We would much rather the model learn to minimise the regret, the *DQ*, of the induced decision, i.e $f(\hat{\boldsymbol{z}}^*; \boldsymbol{y}) - f(\boldsymbol{z}^*; \boldsymbol{y}) = 0$. Having the model learn to minimise regret should result in on average better decisions as in practice $M_{\theta^*} \neq \mathcal{F}$. This flow of DQ information back into the model is shown in the bottom part of Figure 1.

We can define a *DQ* loss function $DQ(\hat{y}, y) = f(Z^*(\hat{y}); y) - f(Z^*(y); y)$ that captures this regret. Substituting $\boldsymbol{\mathcal{L}}$ for DQ in Equation 1 we can formulate a learning process to find the model parameters, θ^{*DQ}, that will cause the model to make predictions that result in decisions closer to optimal decision. However, directly using this loss to train a ML model presents a number of challenges, as a gradient must be passed through the optimisation step. A common solution to this problem is to use a differentiable surrogate in place of the true DQ, enabling gradient descent methods to be used to learn θ^{*DQ}.

a) Surrogate-based DFL: While methods such as smoothing or relaxation can be used to create differentiable surrogates of the optimisation step [6], doing so requires domain expertise. Berthet et al. [20] show that it is possible to differentiate through linear optimisation problems during the training process, applying stochastic perturbations $\hat{y}$ to calculate a smooth gradient. Shah et al. [7] expand on this work, proposing LODL, a pre-learned, per-instance parameterised, differentiable, and convex by construction surrogate model for DQ where $LODL_{\phi_i}(\hat{y}_i, y_i) \rightarrow DQ(\hat{y}_i, y_i)$. By pre-learning, they remove the need to perform optimisation in the training loop and allow the same perturbations to be used to train multiple predictive models. Additionally, their suggested loss models generalise it to arbitrary optimisation problems.

Of the surrogate models proposed by Shah et al., the most generally effective appears to be the Directed Quadratic. Its strength lies in the fact it can capture the effect of correlation different dimensions have on one another. It is defined as:

$$\text{DirectedQuadratic}_{\phi_i}(\hat{y}_i, y_i) = \Delta^T \cdot H \cdot \Delta \tag{3}$$

Where $\phi_i = H, \gamma$ and $\Delta = \hat{y}_i - y_i + \gamma$. We note a key change in our formulation of the model, the inclusion of a learnable bias term γ, initialised, to zero added to the difference between predicted and observed ys. Akin to the original formulation, H is $\dim(y) X \dim(y)$ a matrix with its values conditional on the value of the biased difference, Δ:

$$H_{ij} = \begin{cases} H_{ij}^{++}, & \text{if } \Delta_i \geq 0 \text{ and } \Delta_j \geq 0 \\ H_{ij}^{+-}, & \text{if } \Delta_i \geq 0 \text{ and } \Delta_j < 0 \\ H_{ij}^{-+}, & \text{if } \Delta_i < 0 \text{ and } \Delta_j \geq 0 \\ H_{ij}^{--}, & \text{otherwise} \end{cases}$$

Performing LODL is computationally expensive. The total complexity of learning a surrogate loss model for every instance and then learning the predictive model is:

$$\Theta(N \cdot K \cdot G + N \cdot K \cdot S_{LODL} \cdot T_{LODL}) \tag{4}$$

Where: N is the number of instances; K is the number of perturbations of each instance; G is the time to run the optimiser on one instance; S_{LODL} the number of steps the respective model is trained for; and T_{LODL} the time to run a forward and backwards pass of the respective the model.

IV. Evolve LODL: A More Efficient DFL

Reducing the complexity of DFL is needed for it to scale to large problems. From Equation 4, we consider both Z and T_{LODL} to be constant and reducing N, the number of

instances seen by the predictive model, is likely to reduce the generalisation of the learned predictive model. We propose an alternative perturbation strategy to improve the sample efficiency of learning LODL, enabling the same performance with a reduction of K.

A. Sampling Strategies

In order to train a surrogate loss model in a supervised setting, a labelled training set must be generated. For each problem instance y_i, we generate a set of K perturbations $[y_i^1, \ldots, y_i^K]$ and calculate the labels $DL(y_i^k, y_i)$. Shah et al. use a random sampling strategy "All Perturbed" [7], where the perturbed instance is generated by simply adding zero mean Gaussian noise.

$$y_i^k = y_i + \alpha \cdot \mathcal{N}(0, \sigma(y)) \tag{5}$$

Where α is a tunable hyperparameter, the mutation rate, we set $\alpha = \mathrm{std(Y)}$.

While simple and effective, we note that the distribution of DQ of the generated is skewed towards 0 breaking the i.i.d assumptions of ML. We propose two distinct strategies to generate a set of training examples with a smoother distribution in an attempt to increase sample efficiency.

1) **Resample:** We undersample from a set of λK samples. The samples are grouped into n bins of width h based on their DQ, ($\lambda = 16$, $n = 10$ and, $h = 2\frac{IQR}{\sqrt[3]{n}}$). K samples are randomly drawn, without replacement, evenly sampling from each bin. In the case a bin is exhausted of samples, another set of λK is generated. Refilling is limited to 3 times before a random sample is drawn.
   The resample strategy acts to test our hypotheses that a set of perturbed samples with a flatter distribution of DQ will learn a better surrogate. We note that resampling is not a practical approach due to the redundant optimiser runs, in the worst case $3\lambda - 1$ perturbed samples are evaluated but not used to train the surrogate.
2) **Directed:** We attempt to produce a set of samples with a distribution of DQ similar to the one generated using the resample strategy, without the redundant optimiser runs. We propose using an EA with an objective to maximise the DQ to direct the distribution of perturbations without redundant optimise
   As a population-based search method, an EA successively generates a population, $\mathbf{Y}_g$, of P candidate solutions at generation, g. The population in generation g is denoted as $\mathbf{Y}_g = \{\mathbf{x}_g^1, \ldots, \mathbf{x}_g^P\}$. Each successive generation is an evolution of the population created by the application of genetic operators: selection, crossover, and mutation.
   The mutate operator uses the same noise addition defined in Equation 5. Crossover, is a weighted mean of the two samples $y_1 \cdot n$ and $y_2 \cdot (1 - n)$ where $n \sim \mathcal{U}[0, 1]$. The sampling strategy is described in Algorithm 1. Unlike a traditional EA, there is no elitism, none of the "best" solutions is carried from generation $g - 1$ to g. Each population is generated from a combination of both the best candidates from Y_{g-1}, and Y_0.

Algorithm 1 Directed perturbation for a single instance $\boldsymbol{y_i}$.

Require: $\boldsymbol{\nu}$:= Sample rate, $\boldsymbol{\mu}$:= Reinitialise rate
Require: G := Number of generations
Require: P := Population size
function DIRECTEDPERTURB(y)
  $c \leftarrow \mathbf{DL}(y, y)$ ▷ Initialise regularisation term
  $Y_0 = \{(y_0^0, 0) \ldots (y_0^P, 0)\}$ ▷ The initial population
  for $g := 1$ to G **do**
    $pb \leftarrow \mathbf{best}_{\nu \cdot P}(Y^{g-1})$ ▷ The *best* $\nu \cdot P$ samples from the previous generation
    $pi \leftarrow \mathbf{sample}_{\mu \cdot P}(Y^0)$
    $ps \leftarrow pb \cup pi$
    for $p := 1$ to P **do**
      $p1, p2 \leftarrow \mathbf{sample}_1(ps), \mathbf{sample}_1(ps)$
      $y^\epsilon \leftarrow \mathbf{mutate} \circ \mathbf{crossover}(p1, p2)$
      $Y_g^p \leftarrow y^\epsilon, c - \mathbf{DL}(y^\epsilon, y)$
    end for
  end for
  return $Y_1 \cup Y_2 \ldots \cup Y_G$
end function

V. EXPERIMENTS

To validate our method, we run experiments using two problem domains. Cubic is a standard baseline problem taken from [7]; and Solar, is a real-world problem in solar energy systems. We compare the DQ achieved on each problem, normalised to the true optimum solution.

a) Cubic: Given a feature $x_n \sim \mathcal{U}[-1, 1]$. Learn a linear model $\hat{y_n} = mx_n + b$ to predict y_n where the true value of y_n is defined as a cubic function: $y_n = 10x_n^3 - 6.5x_n$. The model is used to predict a vector of $N = 50$ samples $\hat{y} = [\hat{y_1}, \ldots, \hat{y_N}]$. The optimisation process selects the $B = 1$ samples from the vector $[y_1, \ldots, y_N]$ that has the highest value: $Z^*(y) = \arg\max(y)$.

b) Solar: This problem models a grid-connected building with both solar panels and a battery installed. We aim to minimise the total CO_2 emissions of energy drawn from the grid by scheduling when to charge and discharge the battery.

The predictive model must forecast the next, $H = 10$, periods of household energy demand (kWh) and solar power generation (kWh). We use two separate predictive models: $M_{\theta_s}(x) \rightarrow [\hat{s_1}, \ldots, \hat{s_H}]$, $M_{\theta_d}(x) \rightarrow [\hat{d_1}, \ldots, \hat{d_H}]$. We use a 2 hidden layer NN with 32 units for both predictive models.

Given the carbon cost c ($\frac{\mathrm{g}}{\mathrm{kWh}}$), a household energy demand d (kWh) and, solar generation s (kWh) for the next H periods. The optimiser must create a charge/hold/discharge schedule b for the battery. A positive value in b indicates the battery is charging and a negative value indicates discharging with the battery being used to meet the house's energy demands.

$$\boldsymbol{Z^*}(c, d, s, b) = \arg\min_b \sum_{h=1}^{H} c_h \cdot (d_h - b_h - s_h) \tag{6}$$

There are constraints on the charge / discharge rate ($dr \leq b_h \leq cr$) of the battery. For each period, the battery, B, is modelled

with a maximum capacity of 10 kWh ($0 \leq B_h \leq 10$) and is leaky if it is not charged or discharged:

$$B_{h+1} = \begin{cases} B_h \cdot decay, & \text{if } b_h = 0 \\ B_h + b_h, & \text{otherwise} \end{cases}$$

For the optimisation step, we use real-world data from households across the UK. The data consists of half-hourly smart meter data taken from January 2017 to December 2018 [23]. This data was combined with the corresponding irradiance [24], to model solar generation and grid carbon cost [25]. For each week households with the top 20% energy use were selected and data was randomly sampled from them.

The input features, x are generated synthetically by passing the target vectors c and d through a randomly initialised 1-layer NN generating pseudo-random features c^ϵ and d^ϵ. Using synthetic data allows us to control the difficulty of the learning problem by adjusting the strength of the correlation between the feature and target vectors.

A. Results

Table I shows the best results achieved in each of the problem domains using our two perturbation strategies. We compare against the standard 2-stage approach where the model was trained using $\mathcal{L}(\hat{y_i}, y) = (\hat{y_i} - y)^2$ and the random perturbation strategy. We can see that the LODLs work as expected, with all three strategies improving over the 2-stage approach. We also note that all three strategies converge, meaning that regardless of the strategy used, if given enough samples it is possible to learn effective LODL surrogates.

TABLE I: Normalised DQ of the best results for each perturbation strategy on each domain. Higher is better.

Perturbation	Problem	
	Cubic	Solar
2-stage	-0.93 ± 0.00	0.28 ± 0.01
Random	0.96 ± 0.00	0.32 ± 0.01
Resample	0.96 ± 0.00	0.34 ± 0.01
Evolve	0.96 ± 0.00	0.33 ± 0.01

Table II and Table III show how the DQ varies as we adjust the number of samples used to train the surrogate model, K, in the Cubic and Solar domains, respectively. For the Cubic problem, we sample between 2^7 - 2^{10} as we found values of K higher than 2^{10} all three strategies were able to learn surrogates that converged to the best possible predictive model. In the Solar domain, we used between 2^9 - 2^{13} samples.

In both the Cubic and Solar domains, our evolved and resampled perturbation strategies were able to converge on the best model using fewer samples than purely random perturbations. This supports our hypothesis that a more diverse distribution of samples can learn a better surrogate with fewer samples. This would suggest that directing the sampled perturbations of each instance to maximise the DQ generates a more representative training set to train the surrogate model.

TABLE II: The DQ learned using each of the sampling strategies. Higher is better with the best result in bold unless all results are equal. (Cubic)

Perturbation	Number of Samples			
	1024	512	256	128
Random	0.96 ± 0.00	0.58 ± 0.76	-0.93 ± 0.00	-0.93 ± 0.00
Resample	0.96 ± 0.00	**0.96 ± 0.00**	-0.93 ± 0.00	-0.93 ± 0.00
Evolve	0.96 ± 0.00	**0.96 ± 0.00**	**-0.18 ± 0.93**	-0.93 ± 0.00

B. Runtime Complexity

The result from the Cubic domain indicates that the use of evolved perturbations yields equivalent performance to random with $\frac{1}{2}$ the number of samples. In the Solar domain, the difference is even greater with evolved perturbations producing the same performance with $\frac{1}{4}$ the number of samples compared to random.

We found that when training DFL models using LODL surrogates, the bulk of the computation is used running the solver $DL(y_i^k, y_i)$ to generate the datasets to learn the surrogate models. Thus, the overhead of running the EA was negligible. This also means there was a linear relationship between K and the wall clock runtime. Table IV shows the runtime to generate the perturbed dataset for each of the strategies. We can see that there is no difference between random and evolve.

C. Sampling Distributions

This section analyses the distribution of DQ generated by each of the sampling strategies.

Figure 2a shows the DQ distribution of $K = 4096$ perturbations of 5 instances of the solar problem. We can clearly see that the random perturbation has a much stronger peak close to $\Delta DQ = 0$. The resample approach, as expected, produced a much flatter distribution with more samples beyond the tail of the random distribution. The evolve distribution sits between the 2, with a noticeable peak close to $\Delta DQ = 0$ but also more samples beyond the tail of random, similar to the resample strategy. The evolved sampling strategy generates a set of perturbed samples that generate a more evenly distributed DQ space. This enables effective surrogate losses to be learned with fewer samples.

In Figure 2b we can see the evolution of the population generated using the directed perturbations at each generation. While the early generations closely mirror the distribution of the random perturbations, with the majority of perturbations resulting in a $\Delta DQ = 0$. Future generations yield perturbations that result in more extreme ΔDQ. Furthermore, we note the double peak in later generations, this is due to the reinitialisation step, where we resample from the unperturbed instances. We found this step to be necessary to prevent the exploration from getting stuck in local optima or evolving perturbations that lead to unbounded error (such as continuously increasing the solar values).

TABLE III: Normalised DQ learned using each of the sampling strategies for the cubic problem domain. Higher is better, with the best result in bold. (Solar)

Perturbation	Number of Samples				
	8192	4096	2048	1024	512
Random	0.32 ± 0.01	0.31 ± 0.01	0.31 ± 0.01	0.30 ± 0.01	0.29 ± 0.01
Resample	-	**0.34 ± 0.01**	**0.33 ± 0.01**	**0.32 ± 0.02**	**0.32 ± 0.02**
Evolve	**0.33 ± 0.01**	0.33 ± 0.01	0.32 ± 0.01	**0.32 ± 0.01**	0.31 ± 0.01

TABLE IV: Approximate runtime to generate a set of K samples for each problem domain. For Cubic $K = 1024$ and Solar $K = 8192$. All experiments were performed using an A6000 GPU, 8 CPU cores and 32 GB of RAM.

Perturbation	Domain	
	Cubic	Solar
Random	10 sec	10 mins
Resample	1 min	1.5 h
Evolve	10 sec	10 mins

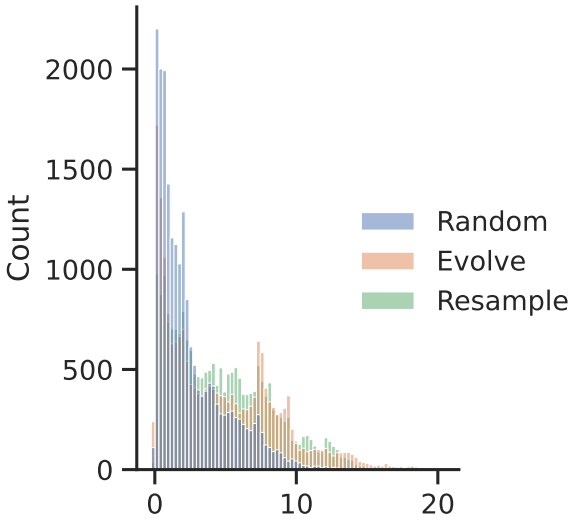
VI. Conclusion

We have proposed a directed perturbation strategy to generate training samples to learn a surrogate loss model for decision-focused learning (DFL) using LODL. We have shown that it is possible to learn an effective surrogate loss model with fewer perturbed samples by employing the directed perturbation strategy. We demonstrated this on two problem domains.

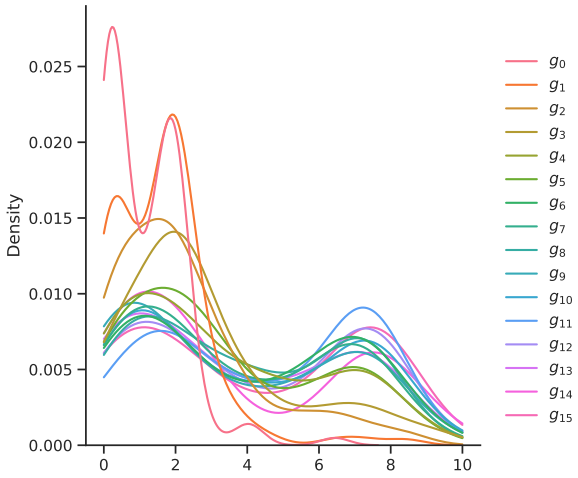
One potential issue with the directed perturbation strategy is that tuning any hyperparameters to a specific problem requires multiple runs. Additionally, selecting incorrect hyperparameters can cause performance to be worse than standard 2-stage predict-then-optimise. However, future research to estimate the utility of information a perturbation may have on the surrogate can address this by allowing dynamic tuning.

Acknowledgment

The authors would like to thank Elastacloud for providing access to data and computing resources. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. T. Cargan holds a studentship funded by EPSRC and The University of Nottingham. I. Triguero is funded by a Maria Zambrano Senior Fellowship at the University of Granada. This work is also supported by the Spanish Grant PID2023-149128NB-I00 funded by MICIU/AEI /10.13039/501100011033 and by ERDF, EU.



(a) Distribution of the ΔDQ of 4096 perturbations of 5 instances



(b) The distribution of the ΔDQ of samples within the first 16 generations of the evolved perturbations

Fig. 2: An overview of the distribution generated when perturbing 5 instances from the Solar domain with the different sampling strategies.

References

[1] Lawrence E. Jones. *Renewable Energy Integration: Practical Management of Variability, Uncertainty, and Flexibility in Power Grids*. Elsevier Science, 2014. ISBN 0-12-407910-5.

[2] James Kotary, Ferdinando Fioretto, Pascal van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. page 4475 – 4482, 2021.

[3] Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. A survey of contextual optimization methods for decision-

making under uncertainty. *European Journal of Operational Research*, 3 2024. ISSN 03772217. doi: 10.1016/j.ejor.2024.03.020.

[4] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*, volume 21. Springer Berlin Heidelberg, 2018. ISBN 978-3-662-56038-9. doi: 10.1007/978-3-662-56039-6.

[5] Chris Cameron, Jason Hartford, Taylor Lundy, and Kevin Leyton-Brown. The perils of learning before optimizing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36:3708–3715, 6 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i4.20284.

[6] 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. 7 2023.

[7] 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.

[8] Christoph Bergmeir, Frits de Nijs, Abishek Sriramulu, Mahdi Abolghasemi, Richard Bean, John Betts, Quang Bui, Nam Trong Dinh, Nils Einecke, Rasul Esmaeilbeigi, Scott Ferraro, Priya Galketiya, Evgenii Genov, Robert Glasgow, Rakshitha Godahewa, Yanfei Kang, Steffen Limmer, Luis Magdalena, Pablo Montero-Manso, Daniel Peralta, Yogesh Pipada Sunil Kumar, Alejandro Rosales-Pérez, Julian Ruddick, Akylas Stratigakos, Peter Stuckey, Guido Tack, Isaac Triguero, and Rui Yuan. Comparison and evaluation of methods for a predict+optimize problem in renewable energy. 12 2022. URL http://arxiv.org/abs/2212.10723.

[9] Andrew J. Keith and Darryl K. Ahner. A survey of decision making and optimization under uncertainty. *Annals of Operations Research*, 300:319–353, 5 2021. ISSN 0254-5330. doi: 10.1007/s10479-019-03431-8.

[10] Chuan Luo, Bo Qiao, Wenqian Xing, Xin Chen, Pu Zhao, Chao Du, Randolph Yao, Hongyu Zhang, Wei Wu, Shaowei Cai, Bing He, Saravanakumar Rajmohan, and Qingwei Lin. Correlation-aware heuristic search for intelligent virtual machine provisioning in cloud systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:12363–12372, 5 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i14.17467.

[11] Jana Ksciuk, Stefan Kuhlemann, Kevin Tierney, and Achim Koberstein. Uncertainty in maritime ship routing and scheduling: A literature review. *European Journal of Operational Research*, 308(2):499–524, 2023. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2022.08.006.

[12] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. 2009.

[13] Arman Zharmagambetov, Brandon Amos, Aaron Ferber, Taoan Huang, Bistra Dilkina, and Yuandong Tian. Landscape surrogate: Learning decision losses for mathematical optimization under partial information. 7 2023.

[14] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21188–21198. Curran Associates, Inc., 2020.

[15] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *ICLR*, 2022.

[16] 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.

[17] Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68:9–26, 1 2022. ISSN 0025-1909. doi: 10.1287/mnsc.2020.3922.

[18] Adam N. Elmachtoub, Jason Cheuk Nam Liang, and Ryan McNellis. Decision trees for decision-making under the predict-then-optimize framework. *Proceedings of the 37th International Conference on Machine Learning*, 2 2020. doi: 10.5555/3524938.3525206.

[19] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[20] 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.

[21] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1658–1665, 7 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.33011658.

[22] Sanket Shah, Bryan Wilder, Andrew Perrault, and Milind Tambe. Leaving the nest: Going beyond local loss functions for predict-then-optimize. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:14902–14909, 3 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i13.29410.

[23] Direnc Pekaslan, Jose Maria Alonso-Moral, Kasun Bandara, Christoph Bergmeir, Juan Bernabe-Moreno, Robert Eigenmann, Nils Einecke, Selvi Ergen, Rakshitha Godahewa, Hansika Hewamalage, Jesus Lago, Steffen Limmer, Sven Rebhan, Boris Rabinovich, Dilini Rajapasksha, Heda Song, Christian Wagner, Wenlong Wu, Luis Magdalena, and Isaac Triguero. The energy prediction smart-meter dataset: Analysis of previous competitions and beyond. 11 2023.

[24] Met Office. Midas open: Uk hourly solar radiation data, v202207. NERC EDS Centre for Environmental Data Analysis, 2022.

[25] National Grid ESO. Data source - carbon intensity. https://carbonintensity.org.uk/, 2023.