On The Robustness of Decision-Focused Learning

Yehya Farhat

Syracuse University
ymfarhat@syr.edu

Abstract

Decision-Focused Learning (DFL) is an emerging learning paradigm that tackles the task of training a machine learning (ML) model to predict missing parameters of an incomplete optimization problem in which the missing problem parameters are completed by a prediction. DFL trains an ML model in an end-to-end system by integrating the prediction and optimization tasks, providing better alignment of the training and testing objectives. DFL has shown a lot of promise and holds the capacity to revolutionize decision-making in many real-world applications. However, very little is known about the performance of these models under adversarial attacks. We adopt ten unique DFL methods and benchmark their performance under two distinctly focused attacks adapted towards Predict-then-Optimize-based problems. Our results lead us to hypothesize that the models' robustness is highly correlated with the model's ability to find predictions that lead to optimal solutions without deviating from the ground-truth label. Furthermore, we provide insight into how to target the models that do violate this condition and show how these models respond differently depending on the achieved optimality at the end of their training cycles.

1 Introduction

Decision Focused Learning (DFL) is an emerging learning paradigm in the field of *Machine Learning* (ML) that confronts the task of decision-making under uncertainty. The problem of decision-making under uncertainty pertains itself to the sequential task of predicting uncertain quantities using an ML model and then using a *Constrained Optimization* (CO) model to optimize objectives using the predicted quantities.

The conventionally adopted approach to this problem is to consider both the predictive and optimization tasks separately. This *Two-Stage* (TS) approach first involves training an ML model using a traditional loss function such as MSE to map the input features to the relevant parameters of the CO problem. Where afterwards a specialized optimization algorithm is used to solve the CO problem. This approach has obvious advantages, in which the model learning phase is well justified and independent of any secondary task besides finding the best mapping between the input features and labels. In theory, this approach is infallible if we are able to obtain a perfect model to make precise predictions. Indeed if the predictions of the parameters by the trained model are perfectly accurate, they would lead to correct specifications of the CO models which can be solved to yield fully optimal solutions. However, in practice ML models fall short of perfect accuracy which lead these errors propagating to the CO models and which in turn lead to sub-optimal decisions. This put forward the need of modelling the predictive and decision process jointly. DFL is based on training the ML model to make predictions which lead to good decisions. DFl integrates prediction and optimization in an end-to-end system that targets predictions that lead to optimal decision by optimizing a loss function that is based on the resulting decision of the prediction. DFL has been shown to outperform TS on a variety of domains (Mandi et al. 2019; Wilder, Dilkina, and Tambe 2018; Cameron et al. 2021).

The key technical challenge with the DFL paradigm is the gradient-based training of the ML model. Given that the loss needs to be differentiated through the entire prediction and optimization pipeline, a discontinuous mapping is produced when the CO problem operates on discrete variables. To overcome these challenges, many smooth surrogate models for these discrete mappings, along with their differentiation have been proposed. For an extensive survey of the different methodologies, interested readers are refereed to (Mandi et al. 2023)

DFL has garnered increasing attention, but the robustness of these pipelines in adversarial settings remains inadequately understood. Robustness in a model refers to its ability to maintain performance when subjected to adversarial perturbations or novel data inputs. To employ these pipelines effectively in real-world contexts, comprehending their robustness is crucial, as it allows us to evaluate the suitability of the models in various situations. In practical scenarios, discrepancies between training and testing data are common. We analyze a model's robustness by introducing small worst-case perturbations to the input and evaluate their impact on the model's overall downstream performance and its prediction. The The code and data used are accessible through **https://github.com/yehya-**

farhat/AdvAttacks-DFL.git. To the extent of our knowledge, this work is the first to analyze and compare the robustness of the different DFL methodologies under such an adversarial settings.

Overall, this paper makes the following contributions:

- We replicate eleven DFL methods and implement two types of adversarial attacks with proper adaptation to cater towards Predict-then-optimize based problems
- We comprehensively evaluate the DFL methods across three distinct problems and summarize the results
- We offer empirical insights to demonstrate that robustness is directly correlated with the models ability to find the optimal solutions with respect to the ground truth labels

2 Related Work

Model robustness is a crucial concept extensively examined within the realms of deep learning and adversarial machine learning. A multitude of studies have delved deeply into understanding and enhancing model robustness, highlighting the vulnerability of neural networks to specially crafted inputs. *Evasion attacks* represent a methodology where specific inputs are meticulously designed to mislead a neural network, thereby undermining or altering the model's output. Such vulnerabilities have been meticulously documented and analyzed in previous works (Goodfellow, Shlens, and Szegedy 2015; Kurakin, Goodfellow, and Bengio 2017). Another subtle yet potent form of undermining model robustness is through *poisoning attacks*. In this approach, the adversary introduces manipulated data points into the training dataset, intending to degrade the model's overall performance and reliability.

Despite the success and extensive study of evasion attacks, their implementation and performance in non-classification problem settings remains understudied. Regression tasks have no natural margins as in the case of classification tasks that lend themselves to the nice property of having finite output spaces. Adversarial learning in regression settings is hindered with difficulties in defining the the adversarial attacks, its success, and evaluation metrics. Despite these challenges, strides have been made in generalizing evasion attacks for regression problems. (Tong et al. 2018) looked at attacks in a special case of ensemble linear models, althought insightful this linear case doesn't capture the larger class of non-linear regression problems. Other studies have focused on specific applications of regression problems. (Ghafouri, Vorobeychik, and Koutsoukos 2018) Study the regression task in cyber-physical systems by selecting an optimal threshold for each sensor against an adversary. (Deng et al. 2020) Examine the regression task of predicting the optimal steering angle in autonomous driving by using a concept they called an adversarial threshold. They consider an attack successful if the deviation between the original prediction and the prediction of an adversarial example is below the specified threshold. (Meng et al. 2019) Introduce two attacks for specific regression problems in the field of electroencephalogram based brain-computer interfaces and utilize a similar attack success criteria as above. (Gupta et al. 2021) Propose a flexible adversarial attack for regression tasks and analyze its effectiveness using various error metrics. Furthermore, general defenses have been proposed in the context of regression problems. (Nguyen and Raff 2018) introduce a useful defense to reduce the effectiveness of evasion attacks by considering adversarial attacks as a symptom of numerical instability of the learned weight matrix.

In the context of DFL, the exploration of adversarial attacks is an emerging area of study. It has been demonstrated that effective poisoning attacks can be generated against DFL models (Kinsey et al. 2023). Concurrently, a body of work has emerged that focuses on bolstering DFL robustness against label noise. (Johnson-Yu et al. 2023). They cast the learning problem as Stackelberg games and provide bounds on decision quality in the presence of adversarial label drift and propose a learning schema to mitigate this affect by anticipating label drift.

3 Background

3.1 Predict-Then-Optimize Problem Setting

Decisions are often mathematically modeled using CO problems. These problems are well-suited to a wide range of decision-making scenarios. In many real-world applications, certain parameters of the CO problem remain uncertain and require inference from contextual data. The Predict-and-Optimize problem setting involves decision models that are represented by partially defined optimization problems, whose specification is completed by parameters that are predicted from data.

$$x^*(\mathbf{c}) = \arg\min_x \ f(x, \mathbf{c}) \tag{1a}$$

$$\text{s.t.} \quad g(x) \leq 0 \tag{1b}$$

$$h(x) = 0. \tag{1c}$$

The goal of the optimization problem described is to find a minimizer $x^*(\mathbf{c}) \in \mathbb{R}^p$ of the objective function (1a), satisfying the constraints (1b-1c). This paper considers the problem setting where the parameter $\mathbf{c} \in \mathcal{C} \subseteq \mathbb{R}^k$ is unknown and must be inferred as a function of observed empirical features $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^l$. In this context, given a set of past observations pairs $\mathcal{D}_{train} = \{(\mathbf{z_i}, \mathbf{c_i})\}_{i=1}^N$ The predictive task is to learn a parameterized (by w) model $m_w : \mathcal{Z} \to \mathcal{C}$. Such that the model is used to make predictions in the form of $\hat{\mathbf{c}} = m_w(z)$. Subsequently, a decision $x^*(\hat{\mathbf{c}})$ is made based on the predicted parameter. The overarching learning objective is to optimize the set of decisions made over the set of observed features $\mathbf{z} \in \mathcal{Z}$, with respect to some evaluation criterion on those decisions.

3.2 Frameworks for Predict-Then-Optimize

While the ML model m_w is trained to predict $\hat{\mathbf{c}}$, its performance is evaluated on the basis of the corresponding optimal solution of the decision model $x^*(\hat{\mathbf{c}})$. Using standard ML approaches, learning the model m_w can only be supervised by ground-truth $\mathbf{c}$ using a standard loss function $\mathcal{L}(\hat{\mathbf{c}}, \mathbf{c})$, such as mean squared error (MSE). But given that we are concerned with optimizing the decision $x^*(\hat{\mathbf{c}})$, it would be favorable to train m_w to make predictions $\hat{\mathbf{c}}$ that optimize

the the decision. This puts forth two different learning approaches for the model m_w.

Two-Stage (TS) The conventional learning paradigm is to generate accurate parameter predictions $\hat{\mathbf{c}}$ with respect to the ground-truth values $\mathbf{c}$, this Two-Stage approach learns the predictive model by minimizing a standard loss function, such as MSE.

$$\min_{w} \sum_{(z,c)\in\mathcal{D}_{\text{train}}} \|m_w(z) - c\|^2 \tag{2}$$

The final trained model is then used to make a prediction on a new data point $\mathbf{z}$ such that the prediction is used to optimize the objective $x^*(m_w(z))$ in Eq. (1a)

Decision-Focused learning (DFL) By contrast, DFL trains the ML model to optimize the evaluation criteria that measures the quality of the decision.

$$\min_{w} \sum_{(z,c)\in\mathcal{D}_{\text{train}}} f(x^*(m_w(z)), c) - f(x^*(c), c) \tag{3}$$

Similar to the TS approach, after training the model is called to make new predictions on new data points, which are then passed to (1a). The advantage of DFL is the alignment of the training objective and the testing objective f. To optimize the objective in DFL, it is common to use gradient descent. This will require backpropagating through the optimal decision x^*. Depending on the CO problem, The gradient of the optimal decision is not always easily computed. CO problems can be categorized in terms of the form taken by the their objectives and constraints. These forms define the properties of the optimization mapping, such as its continuity and differentiability. This puts forth two major challenges, the first is the optimization problem solution mapping $\hat{\mathbf{c}} \rightarrow \mathbf{x}^*(\hat{\mathbf{c}})$ lacks a closed form which can be differentiated directly. And the second, depending on the problem, a lot of useful optimization problems have a non-differentiable mapping on some points, and a zero-valued gradients on others, making gradient descent unusable. These challenges have put forward the need of developing methodologies which address the challenge of differentiating an optimization mapping for DFL in gradient-based training. Different approaches propose different smoothed surrogate approximations which can be used for backpropagation.

4 Methodology

4.1 Objective

The problem of adversarial attacks is closely related to the robustness issue for a neural network, i.e. its sensitivity to perturbations. Let $m : \mathcal{Z} \rightarrow \mathcal{C}$ denote the trained model, for which either Eq. (2) or Eq. (3) was used for training to produce an output $m(z)$. We define an additive vector $\epsilon \in \mathbb{R}^l$ to perturb the original input and get the perturbed output $m(z + \epsilon)$. Unlike classification problems the output of the model does not have well defined success margins, in the classification scenario we desire no change in the response for small change in the input. For regression problems, we must expect *some* change in the response for any change in the input. In such a setting, attacking the network amounts to finding a perturbation ϵ of preset magnitude which makes the output maximally deviate from a reference output. The reference output may be defined as the model output $m(z)$ or the ground truth label c. In this work we are interested in studying the behavior of the different DFL methodologies under input perturbations, thus we will also adopt the reference output of the decision problem under the model prediction $x^*(m(z))$ and the ground truth label $x^*(c)$. In this context, the measure of deviation and magnitude of perturbation play an important role in the problem formulation. We measure the output deviation using an ℓ_q-norm where $q \in [1, +\infty]$. It is important to underscore that opting for this approach is logical in the case of regression problems. The ℓ_2 and ℓ_1 norms are often applied as loss functions during the training process. Conversely, the $\ell_{+\infty}$ norm is commonly utilized when addressing matters of reliability

Given a robust predictive function $m_r(.)$ and a quality function $f(.)$, we would expect that

$$|f(m_r(z)) - f(m_r(z + \epsilon))| \leq \Delta \tag{4}$$

$$|m_r(z) - m_r(z + \epsilon)| \leq \delta \tag{5}$$

for some threshold Δ and δ. Most adversarial work focuses on attacks that violate this expectation. Making it such that small perturbations cause drastic changes in the output. Its important to note that satisfying both Eq. (4) and Eq. (5) need not be necessary in the context of DFL. DFL focuses on making predictions that lead to good quality decision, regardless of the prediction itself. In cases where the CO problem has non-unique optimal solutions, DFL might allow for the satisfaction of Eq. (4), while simultaneously allowing for the violation of Eq. (5). But in a setting where we have a perfect predictive and robust model (with respect to the ground truth labels) then we would expect both inequalities to be satisfied.

In this work we focus on studying the behavior of Δ and δ with respect to the different DFL methodologies. Our objective is to categorize the sensitivity of each learned model by measuring the variations in output under two distinctly focused attacks. Additionally, we analyze the relationship between the behavior of the downstream task $f(m_r(z + \epsilon))$ and the predictive task $m_r(z+\epsilon)$, aiming to understand how perturbations affect both tasks.

4.2 Problem Sets

Problem description We select 3 diverse problems for benchmarking: Warcraft shortest path, Portfolio optimization, and Knapsack. All of which have been previously used as benchmarks in the DFL literature and their datasets are publically available. All these problems pose both a predictive and optimization task.

Warcraft Shortest Path This problem was adopted from the work of (P. et al. 2019), utilizing the openly accessible warcraft terrain map images dataset (guyomarch 2017). It features images configured as a $d \times d$ grid, with each grid cell—or pixel—assigned a specific cost that requires prediction. The objective is to then identify the shortest path from the top left-pixel to the bottom-right pixel. Unless situated

on the grid's boundary, one may proceed in any of eight possible directions from a given pixel, framing the challenge as a node-weighted shortest path problem on a graph with d^2 vertices and up to d^2 edges. This problem is transformed into the more conventional edge-weighted shortest path problem by dividing each weighted node into a pair of nodes, with the original node's weight transferred to the edge that connects these new nodes.
The problem of shortest path can be formulated as an LP problem with the following form:

$$\min_x \ c^\top x \tag{6a}$$
$$\text{s.t.} \quad Ax = b \tag{6b}$$
$$0 \leq x \tag{6c}$$

Where $A \in \mathbb{R}^{|V|\times|E|}$ is the incidence matrix of the graph. $x \in \mathbb{R}^{|E|}$ is a binary vector whose entries are 1 if the corresponding edge is selected and 0 otherwise. $b \in \mathbb{R}^{|V|}$ is a vector whose entries are all 0 except the entries that corresponding to the source and sink node where are 1 and -1, respectively.

The predictive task involves the use of a *convolutional neural network* (CNN) to determine the cost associated with each node (pixel), with costs ranging from 0.8 to 9.2 based on the pixel's visible features. The model processes the $d \times d$ image to output the costs for all d^2 pixels.

Portfolio Optimization This problem was adopted from the work of (Elmachtoub and Grigas 2020). The problem entails predicting asset prices based on empirical data, subsequently a risk constrained optimization problem is solved to obtain a portfolio that maximizes expected return. The problem formulation of Portfolio optimization can expressed as the following:

$$\max_x \ c^\top x \tag{7a}$$
$$\text{s.t.} \quad x^\top \Sigma x \leq \lambda \tag{7b}$$
$$\mathbf{1}^\top x \leq \mathbf{1} \tag{7c}$$
$$0 \leq x \tag{7d}$$

The Synthetic input-target pairs $(\mathbf{z}, \mathbf{c})$ are randomly generated according to a random function with a specified degree of nonlinearity, we refer to this as $Deg \in \mathbb{N}$. A detailed breakdown of the random function used for input-target generation can be found in the appendix.
$\mathbf{1}$ represents a vector composed entirely of ones. Σ refers to a pre-established covariance matrix between asset returns. The predictive task utilizes a simple linear neural network model, whose inputs is a feature vector $\mathbf{z} \in \mathbb{R}^l$ and output is the return vector $\mathbf{c} \in \mathbb{R}^k$ that represents asset prices.

Knapsack This experiment setup was adopted from the work of (Mandi et al. 2019). The problem involves solving the knapsack problem's objective, which entails selecting a subset of items with maximum value from a specified set, while adhering to the capacity limitation. The problem formulation can be written as follows:

$$\max_x \ c^\top x \tag{8a}$$
$$\text{s.t.} \quad w^\top x \leq Capacity \tag{8b}$$
$$x \in \{0, 1\} \tag{8c}$$

The weights of all items and the capacity constraint are known, hence The prediction task is to predict the value of each item. The problem dataset utilizes the *Irish Single Electricity Market Operator* (SEMO) (Ifrim, O'Sullivan, and Simonis 2012). In this arrangement, each day is treated as an individual optimization problem, with every half-hour representing an item in the knapsack. Consequently, both the cost vector $\mathbf{c}$ and the weight vector $\mathbf{w}$ consist of 48 elements, each corresponding to a half-hour segment. Each item in the cost array is associated with an 8-dimensional feature vector. This vector represents various attributes, encompassing elements like weather conditions, projected energy load, and actual wind speed. The weight array remains constant throughout and is synthetically generated, as previously established by (Mandi et al. 2019). A detailed breakdown of the synthetic weight vector generation can be found in the appendix. As the previous problem, the predictive task entails a linear model, whose input is a feature vector $\mathbf{z} \in \mathbb{R}^{48\times 8}$ and output vector $\mathbf{c} \in \mathbb{R}^{48}$.

4.3 Attacks

Fast Gradient Sign Method: Prediction-Focused The Fast Gradient Sign Method (FGSM) is a L_∞ bounded attack (Goodfellow, Shlens, and Szegedy 2015). Given a training data point (z, c), a cost function J, and the model parameters w, the adversarial data point $\tilde{z}$ is computed as follows

$$\tilde{z} = z + \epsilon sign(\nabla_z J(w, z, c))$$

where the cost function J is a standard loss that calculates the error of the prediction with respect to the ground truth labels $\mathbf{c}$, and where ϵ is the attack magnitude.

Fast Gradient Sign Method: Decision-Focused Similar to the Prediction-Focused FGSM attack implementation, We implement a slightly modified version of the attack by opting for a different cost function. Let $\acute{J}$ represent the decision cost function of the prediction with respect to the ground truth-decision $x^*(c)$ (Regret). Similarly, we calculate the adversarial data point $\tilde{z}$ as

$$\tilde{z} = z + \epsilon sign(\nabla_z \acute{J}(w, z, c))$$

Its important to note that the calculation of $\nabla_z \acute{J}(.)$ is not always straightforward, as it will run into the same issues of non-differentiability, and zero-valued gradients. To that end, we will need to instead utilize the smoothed surrogate approximations of each respective end-to-end DFL pipeline to compute this attack.

Attack Setting In our experiments, we assess the performance of the prediction-focused FGSM (PF) and decision-focused FGSM (DF) attacks by varying the attack magnitude parameter ϵ, at levels of $0.01, 0.1$, and 0.15, respectively. For each respective DFL methodology, we adopt the same approximation used in training to allow for useful gradient computations of $\nabla_z \acute{J}(.)$.

Table 1: Overview of the different DFL methodologies adopted for experimentation

Methodology	Problem Form
Quadratic Programming Task Loss (QPTL) (Wilder, Dilkina, and Tambe 2018)	LPs, ILPs
Differentiation of Blackbox Combinatorial Solvers (DBB) (P. et al. 2019)	Linear Objective
Fenchel-Young loss (FY) (Blondel, Martins, and Niculae 2020)	Linear Objective
Implicit maximum likelihood estimation (IMLE) (Niepert, Minervini, and Franceschi 2021)	Linear Objective
Interior Point Solving Method (IntOpt) (Mandi and Guns 2020)	LPs, ILPs
Smart Predict Then Optimize (SPO) (Elmachtoub and Grigas 2020)	Linear Objective
Maximum A Posteriori approximation (MAP) (Mulamba et al. 2021)	General optimization problems
Pairwise Learning to Rank (Pairwise) (Mandi et al. 2022)	General optimization problems
Pairwise Learning to rank Difference (Pairwise Diff) (Mandi et al. 2022)	General optimization problems
Listwise Learning to rank (Listwise) (Mandi et al. 2022)	General optimization problems

4.4 DFL methodologies

We adopt 10 different DFL methodologies and the two-stage method as our baseline for comparison. All mentioned DFL techniques are an attempt to overcome the challenge of differentiating the optimization mapping by proposing different smoothed surrogate approximations of $\frac{d\mathcal{L}(x^*(\hat{c}))}{d\hat{c}}$ or $\frac{dx^*(\hat{c})}{d\hat{c}}$. Table 1 provides an overview of the different methods and their form adopted for comparison.

4.5 Network Architecture and Training

The network architecture for the Warcraft shortest path is a CNN. As in (Mandi et al. 2023) we adapted the ResNet18 architecture, which includes the first five layers of ResNet18 followed by a max-pooling operation that assists in predicting the underlying cost for each pixel, and a Relu activation function to ensure positive edge weights. In addressing the Knapsack and Portfolio problems, we employ a single-layer feed-forward neural network, devoid of hidden layers (A linear model). The rational behind utilizing such a simple predictive model lies in evaluating the effectiveness of the DFL methods under conditions where predictions lack high accuracy. We utilize Adam as our optimization algorithm (Kingma and Ba 2017) and ReduceLROnPlateau learning rate scheduler (Paszke et al. 2019). We utilize Cvxpylayers (Agrawal et al. 2019) as our solver for QPTL, QP problems. For the other methods we employ Gurobi (Gurobi Optimization, LLC 2023) and OR-Tools (Perron and Furnon 2023) as our solvers. The best hyperparameter selection of each model for each experiment can be found in the appendix.

4.6 Evaluation Metrics

To understand and analyze the performance of the DFL methodologies under the proposed adversarial attacks, we train the models using the best respective hyperparameters combinations for each test problem, 10 trails of all the methods are run on the test dataset. Unless otherwise mentioned the evaluation is done based on the 4 error metrics in Table 2. In practical scenarios, there can be instances where the vectors $\mathbf{c}$ or $\hat{\mathbf{c}}$ have multiple non-unique solutions. Consequently, the machine learning model tasked with predicting the cost vector may exhibit undesirable behavior at the beginning of training by predicting all cost parameters to be zero. In such situations, although the resulting solution might technically be optimal and feasible, the predicted cost vector loses its meaningfulness, rendering the solution less useful or interpretable. Instead, we operate under the assumption that the optimal solution $x^*(c)$ is derived through the use of an optimization oracle. In cases where non-unique solutions exist, ties are broken in a prespecified manner to return a single optimal solution.

5 Results and Discussion

In the next section, we assess the average performance of 11 methods in 10 trials against various adversarial attacks. The primary objective of our research is to examine the overall robustness of these models. Consequently, when we mention the 'best' models, we refer to those demonstrating the least deviation from the initial solution when subjected to both types of attacks. Note that for the Two-Stage method, we only present metrics related to the Prediction-Focused attack given that the model does not have a direct surrogate decision loss function associated with it.

5.1 Knapsack Results

Two instances of the knapsack problem are under consideration, each associated with a different capacity: 60 and 120. The line plot of relative regret error with respect to the varying attack magnitudes of the problem instance with a capacity of 120 is presented in figures 1. The generated box plots for the four metrics in Table. 2 and the rest of the line plots can be found in the appendix.

With a capacity of 120, the models exhibiting the best overall robustness on decision quality, are QPTL, TS, DBB, and IMLE. SPO does relatively well under the PF attack but is not able to hold the same performance under the DF attack. Under prediction quality metrics, the best overall performing models are FY, listwise, pairwise, and MAP. With a capacity of 60, the best-performing models in terms of decision quality are DBB and IMLE. FY performs well under the DF attack but poorly under the PF attack. SPO faces a similar issue as observed in the 120-capacity instance: it performs poorly under the DF attack but well under the PF

Mean Accuracy Error (MAE) =	$\frac{1}{\mathcal{D}_{\text{test}}} \sum_{i=1}^{\mathcal{D}_{\text{test}}} \|m(z_i + \epsilon) - c_i\|_q$
Fooling Error (FE) =	$\frac{1}{\mathcal{D}_{\text{test}}} \sum_{i=1}^{\mathcal{D}_{\text{test}}} \|m(z_i + \epsilon) - m(z_i)\|$
Relative Regret Error (RRE) =	$\frac{1}{\mathcal{D}_{\text{test}}} \sum_{i=1}^{\mathcal{D}_{\text{test}}} \frac{c_i^\top (x^* m(z_i + \epsilon) - x^*(c_i))}{c_i^\top x^*(c_i)}$
Fooling Relative Regret Error (FRRE) =	$\frac{1}{\mathcal{D}_{\text{test}}} \sum_{i=1}^{\mathcal{D}_{\text{test}}} \left\| \frac{c_i^\top (x^* m(z_i + \epsilon) - x^*(c_i))}{c_i^\top x^*(c_i)} - \frac{c_i^\top (x^* m(z_i) - x^*(c_i))}{c_i^\top x^*(c_i)} \right\|$

Table 2: Error Metrics Considered for Evaluation

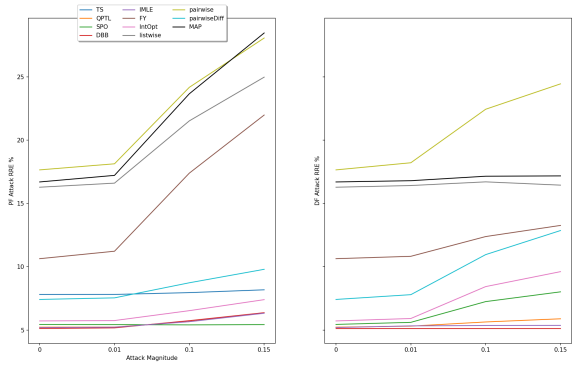

Figure 1: Knapsack capacity 120 RRE, PF Attack left, DF attack right

attack. Regarding prediction metrics, the best models are FY, Listwise, Pairwise, and MAP.

There are interesting observations to note on this problem. The first is that the problem clearly has non-unique optimal solutions, as evidenced by all the models exhibiting high MAE errors while tending toward an optimal decision error. The second observation concerns the varied behavior of these models in terms of robustness when subjected to different types of attacks. The models with the highest initial RRE are the least robust under the PF attack while the models with the lowest initial RRE are the most robust under the PF attack. This behaviour is reversed when the models are exposed to the DF attack. Under the DF attack the models exhibiting the lowest initial RRE are the least robust, while the models with highest initial RRE become the most robust. We believe this is due to what the learned models accomplish at the end of their training cycles and how the different attacks uniquely interact with each method. By definition the models with the highest RRE have not learned how to make good predictions with respect to the decision. Thus, When these models are faced with attacks that try and maximize prediction error the attack results in the model outputting predictions with no consideration for the decision quality. On the other hand the models with low initial RRE are able to maintain their decision quality due to them having learned how to make good predictions with respect to the decision, irrespective of worst case perturbation with the aim of maximizing prediction error. We hypothesize that the reversal of this behaviour under the DF attack is due to the methods ability to find the optimal solution. When faced with the DF attack most of the models exhibiting the best robustness on the PF attack become the least robust. Assuming that the optimal solution has been found by those methods, maximizing the decision error becomes more feasible. While on the other hand, the models that have not been able to find the optimal solution exhibit relatively small change in decision quality under the DF attack, due the fact that these models have a worse sense of where the optimal solution is.

5.2 Portfolio Results

We examine two versions of the Portfolio optimization problem: Degrees 1 and 16, with a noise magnitude parameter set at 1. It's important to note that the IntOpt method is un-

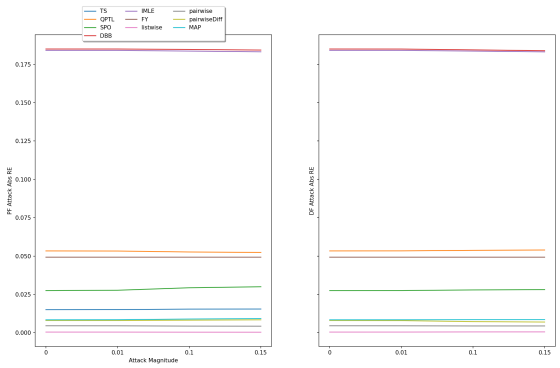

Figure 2: Portfolio deg 16 Absolute RE, PF Attack left, DF attack right

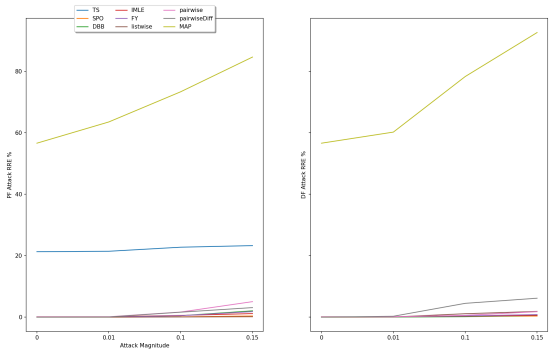

Figure 3: WarCraft 24 $\times$ 24 image size RRE, PF Attack left, DF attack right

suitable in this case because the problem involves quadratic constraints, which the IntOpt method cannot handle. In this problem, for some instances, all the return values are negative, making portfolio optimization with zero return optimal. In such cases, the RRE metric is undefined, as the denominator is zero. Hence, for this problem set, we instead report the absolute regret: $c^\top(x^*(m(z)) - x^*(c))$. We present the line plot of the degree 16 problem instance for the Absolute RE with respect to the attack magnitudes in figures 2. The rest of the line plots and all the box plots can be found in the appendix For this problem, all models perform relatively well, with very negligible variation in decision quality. In such a case the Absolute FRE provides a better glimpse into the minor changes under different attack magnitudes.

For the Degree 16 problem instance, the models that exhibit the least amount of variation under all perturbation levels in terms of decision quality are FY and Listwise. With respect to prediction quality, the best-performing models are FY, Listwise, and MAP. For the Degree 1 problem instance, the best-performing models across all perturbation levels in terms of decision quality are DBB, IMLE, and FY. With respect to prediction quality, the best models are FY and MAP, with FY exhibiting a notably large interquartile range.

In addressing this problem, we observe an interesting phenomenon: the models with low RE do not demonstrate the expected (As in the knapsack problem) poor robustness when subjected to the DF attack. All models maintain high levels of robustness against both types of attacks. This resilience, we believe, stems from the models successfully identifying the optimal solution based on the ground truth label c. This corroborated by the fact that all models consistently exhibit very low MAE, effectively preserving their performance under both attack scenarios.

5.3 Warcraft Shortest Path

The evaluation of the warcraft shortest path problem for image sizes of 12×12 and 24×24 are presented in the following section. We present the line plot of the RRE with respect to the attack magnitudes for the 24×24 problem instance in figure 3. The box plots and the rest of the line plots can be found in the appendix. We note, IntOpt and QPTL necessitate the problem's formulation as a Linear Program (LP) and the use of a primal-dual solver. In our experiment, the predictive machine learning model, specifically a CNN, is tasked with predicting the cost associated with each pixel. Training this ML model proves to be a formidable task due to its extensive parameter set. Consequently, integrating this model with computationally demanding components like an interior point optimizer introduces substantial challenges. Owing to these computational constraints, we were unable to conduct experiments using IntOpt and QPTL.

On the Warcraft problem all models exhibit relatively good robustness. On the 24×24 Image size instance, the best overall performing models under decision quality are SPO, IMLE and DBB. Under prediction quality the best models are FY and MAP. On the 12×12 problem instance, the best models are the same as the 24×24 problem instance. Under decision quality SPO, DBB, and IMLE are overall the best performers while under prediction quality the best overall models are FY and MAP.

We notice a parallel with the portfolio problem in our observations: all models demonstrate commendable robustness against both types of attacks while maintaining a low MAE. This lends credence to our hypothesis that a model's decision robustness is directly linked to its ability to identify the optimal solution in relation to the ground truth label c. This is particularly evident in the performance of the MAP method. It emerges as the model most adversely affected under both adversarial attacks and simultaneously records the highest MAE, thereby reinforcing our hypothesis.

6 Conclusions

In this research, we propose a hypothesis centered around the decision quality robustness of DFL models: their robustness is intricately linked to the model's capacity to discern the optimal solution in alignment with the ground truth label. Our empirical evidence suggests a notable pattern: when models generate predictions that lead to optimal solutions but deviate from ground-truth labels, they tend to be more vulnerable to evasion attacks. Additionally, in examining the behavior of models that identify optimal solutions deviating from the ground-truth labels, we show how their spe-

cific optimality characteristics influence their susceptibility to differently targeted attacks. Our results reveal that such models respond differently to various attack strategies, each uniquely exploiting their particular form of optimality.

Acknowledgments

Special thanks to Dr. Ferdinando Fioretto for his direction and support of this work, and to James Kotary for the interesting conversations and insights.

References

Agrawal, A.; Amos, B.; Barratt, S. T.; Boyd, S. P.; Diamond, S.; and Kolter, J. Z. 2019. Differentiable Convex Optimization Layers. *CoRR*, abs/1910.12430.

Blondel, M.; Martins, A. F. T.; and Niculae, V. 2020. Learning with Fenchel-Young Losses. arXiv:1901.02324.

Cameron, C.; Hartford, J.; Lundy, T.; and Leyton-Brown, K. 2021. The Perils of Learning Before Optimizing. arXiv:2106.10349.

Deng, Y.; Zheng, X.; Zhang, T.; Chen, C.; Lou, G.; and Kim, M. 2020. An Analysis of Adversarial Attacks and Defenses on Autonomous Driving Models. arXiv:2002.02175.

Elmachtoub, A. N.; and Grigas, P. 2020. Smart "Predict, then Optimize". arXiv:1710.08005.

Ghafouri, A.; Vorobeychik, Y.; and Koutsoukos, X. 2018. Adversarial Regression for Detecting Attacks in Cyber-Physical Systems. arXiv:1804.11022.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

Gupta, K.; Pesquet-Popescu, B.; Kaakai, F.; Pesquet, J.-C.; and Malliaros, F. D. 2021. An Adversarial Attacker for Neural Networks in Regression Problems. In *IJCAI Workshop on Artificial Intelligence Safety (AI Safety)*. Montreal/Virtual, Canada.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

guyomarch. 2017. War2Edit. https://github.com/war2/war2edit.

Ifrim, G.; O'Sullivan, B.; and Simonis, H. 2012. Properties of Energy-Price Forecasts for Scheduling. In Milano, M., ed., *Principles and Practice of Constraint Programming*, 957–972. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-33558-7.

Johnson-Yu, S.; Finocchiaro, J.; Wang, K.; Vorobeychik, Y.; and Taneja, A. 2023. Characterizing and Improving the Robustness of Predict-Then-Optimize Frameworks.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kinsey, S. E.; Tuck, W. W.; Sinha, A.; and Nguyen, T. H. 2023. An Exploration of Poisoning Attacks on Data-Based Decision Making. In Fang, F.; Xu, H.; and Hayel, Y., eds., *Decision and Game Theory for Security*, 231–252. Cham: Springer International Publishing. ISBN 978-3-031-26369-9.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial examples in the physical world. arXiv:1607.02533.

Mandi, J.; Bucarey, V.; Mulamba, M.; and Guns, T. 2022. Decision-Focused Learning: Through the Lens of Learning to Rank. arXiv:2112.03609.

Mandi, J.; Demirovic, E.; Stuckey, P. J.; and Guns, T. 2019. Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems. *CoRR*, abs/1911.10092.

Mandi, J.; and Guns, T. 2020. Interior Point Solving for LP-based prediction+optimisation. *CoRR*, abs/2010.13943.

Mandi, J.; Kotary, J.; Berden, S.; Mulamba, M.; Bucarey, V.; Guns, T.; and Fioretto, F. 2023. Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities. arXiv:2307.13565.

Meng, L.; Lin, C.-T.; Jung, T.-R.; and Wu, D. 2019. White-Box Target Attack for EEG-Based BCI Regression Problems. arXiv:1911.04606.

Mulamba, M.; Mandi, J.; Diligenti, M.; Lombardi, M.; Bucarey, V.; and Guns, T. 2021. Contrastive Losses and Solution Caching for Predict-and-Optimize. arXiv:2011.05354.

Nguyen, A. T.; and Raff, E. 2018. Adversarial Attacks, Regression, and Numerical Stability Regularization. arXiv:1812.02885.

Niepert, M.; Minervini, P.; and Franceschi, L. 2021. Implicit MLE: Backpropagating Through Discrete Exponential Family Distributions. *CoRR*, abs/2106.01798.

P., M. V.; Paulus, A.; Musil, V.; Martius, G.; and Rolínek, M. 2019. Differentiation of Blackbox Combinatorial Solvers. *CoRR*, abs/1912.02175.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR*, abs/1912.01703.

Perron, L.; and Furnon, V. 2023. OR-Tools.

Tong, L.; Yu, S.; Alfeld, S.; and Vorobeychik, Y. 2018. Adversarial Regression with Multiple Learners. arXiv:1806.02256.

Wilder, B.; Dilkina, B.; and Tambe, M. 2018. Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization. arXiv:1809.05504.

A Synthetic Data generation Details

A.1 Portfolio Optimization

The Synthetic input-target pairs $(\mathbf{z}, \mathbf{c})$ are randomly generated according to a random function with a specified degree of nonlinearity $Deg \in \mathbb{N}$. The procedure for generating the data is as follows: For a number of assets d and input features of size p. The input samples $x_i \in \mathbb{R}^p$ are independently and identically distributed, each element sampled from a standard normal distribution $N(0, 1)$. A random matrix $B \in \mathbb{R}^{d \times p}$, whose elements $B_{ij} \in \{0, 1\}$ are drown i.i.d. Bernoulli distributions with a probability of 0.5 for the

value 1. For a selected noise level η, $L \in \mathbb{R}^{n\times 4}$ whose entries are drawn uniformly over $[-0.0025\eta, 0.0025\eta]$ is generated. Asset returns are calculated first in terms of their conditional mean as:

$$\bar{c_{ij}} := (\frac{0.05}{\sqrt{p}}(B\mathbf{z_i})_j + (0.1)^{\frac{1}{Deg}})^{Deg}$$

The observed return vectors $\mathbf{c_i}$ are defined as $c_{ij} := \bar{r}_i + Lf + 0.01\eta\xi$, where $f \sim N(0, I_4)$ and noise $\xi \sim N(0, I_d)$. This results c_{ij} to obey the covariance matrix $\Sigma := LL^\top + (0.01\xi)^2 I$ which is used to form our problem constraint and bound the risk, which we define as $\lambda := 2.25e^\top \Sigma e$ where is a constant vector that represents the equal allocation solution. The value of the noise magnitude η is set at 1. We assume that the covariance matrix of the assest returns does not depend on the features. The values Σ and λ are constant and randomly generated for each setting.

A.2 Knapsack

The dataset, derived from the Irish Single Electricity Market Operator (SEMO) is structured such that each day represents an optimization case, and every half-hour is equivalent to an item in a knapsack problem. Accordingly, both the cost vector c and weight vector w have 48 elements, each representing a half-hour. Every cost vector element is linked to an 8-dimensional feature vector. The weight vector remains constant, and its values are synthetically generated using the same approach as in (Mandi et al. 2019) Each of the 48 half-hour periods is assigned a weight w_i by choosing from the set 3, 5, 7. To create a correlation between the weights of the items and their values, the energy price vector is multiplied by the weight vector. This is further randomized by adding Gaussian noise $\xi \sim N(0, 25)$, resulting in the final item values c_i. The total weight for each instance is 240.

B Hyperparameter Selection

The hyperparameter selection for each method and experiment are selected from (Mandi et al. 2023). The authors find the best results through grid search. In this section we provide the list of the used hyperparameters for reproducibility.

Capacity	**60**	**120**
PF (lr)	0.5	1.
SPO (lr)	0.5	1.
DBB (lr, λ)	(0.5, 0.1)	(1., 1.)
IMLE (lr, λ, ϵ, κ)	(0.5, 0.1, 0.5, 5)	(0.5, 0.1, 0.1, 5)
FY (lr, ϵ)	(1., 0.005)	(1., 0.5)
IntOpt (lr, μ, damping)	(0.5, 0.01, 10.)	(0.5, 0.1, 10.)
QPTL (lr, μ)	(0.5, 10.)	(0.5, 1.)
Listwise (lr, τ)	(1., 0.001)	(1., 0.001)
Pairwise (lr, Θ)	(0.5, 10.)	(0.5, 10.)
PairwiseDiff (lr)	1.	1.
MAP (lr)	1.	1.

Table 3: Optimal Hyperparameter Combination for Knapsack

Image Size	**12**	**24**
PF (lr)	0.001	0.001
SPO (lr)	0.005	0.005
DBB (lr, λ)	(0.001, 10.)	(0.001, 100.)
IMLE (lr, λ, ϵ, κ)	(0.001, 10., 0.05, 50)	(0.001, 10., 0.05, 50)
FY (lr, ϵ)	(0.01, 0.01)	(0.01, 0.01)
Listwise (lr, τ)	(0.005, 0.5)	(0.005, 0.5)
Pairwise (lr, Θ)	(0.01, 0.1)	(0.01, 0.1)
PairwiseDiff (lr)	0.005	0.005
MAP (lr)	0.005	0.005

Table 4: Optimal Hyperparameter Combination for Warcraft shortest path

Deg	**1**	**16**
PF (lr)	0.01	0.05
SPO (lr)	0.5	0.5
DBB (lr, λ)	(1., 0.1)	(1., 0.1)
IMLE (lr, λ, ϵ, κ)	(0.5, 0.1, 0.1, 5)	(0.5, 0.1, 0.05,5)
FY (lr, ϵ)	(0.1, 0.01)	(1., 2.)
QPTL (lr, μ)	(0.1, 10.)	(0.05, 10.)
Listwise (lr, τ)	(0.1, 0.01)	(0.05, 0.005)
Pairwise (lr, Θ)	(0.01, 0.01)	(0.1, 0.05)
PairwiseDiff (lr)	0.1	0.05
MAP (lr)	0.01	1.

Table 5: Optimal Hyperparameter Combination for portfolio optimization

C Appendix All Results

In the main text, the line plots for one problem instance and RRE metric is presented in the main text to save space. In this section we present all the of the results and plots.

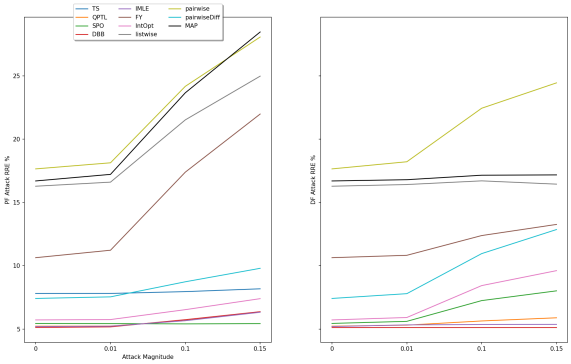

Figure 4: Knapsack capacity 120, RRE

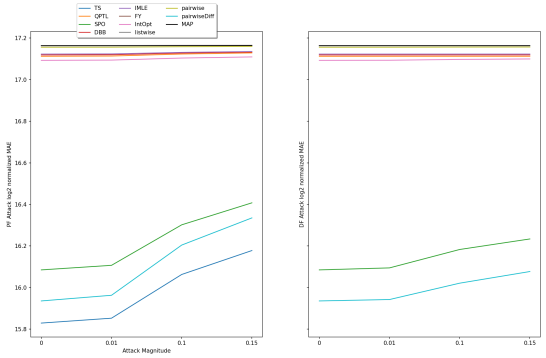

Figure 5: Knapsack capacity 120, MAE

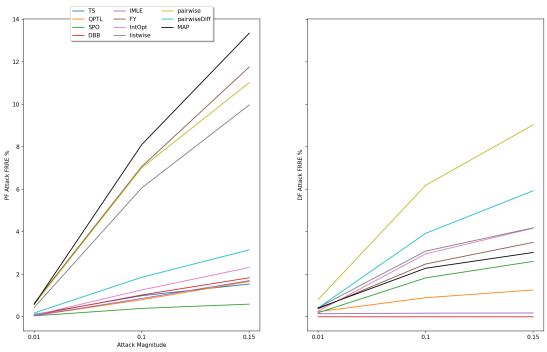

Figure 6: Knapsack capacity 120, FRRE

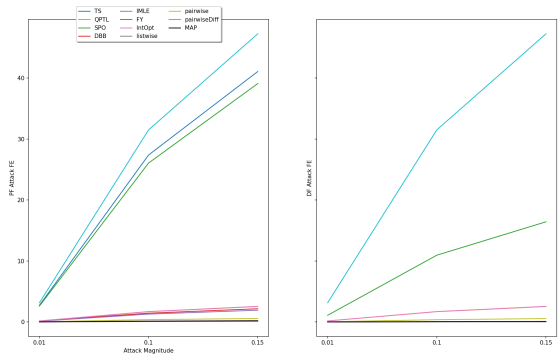

Figure 7: Knapsack capacity 120, FE

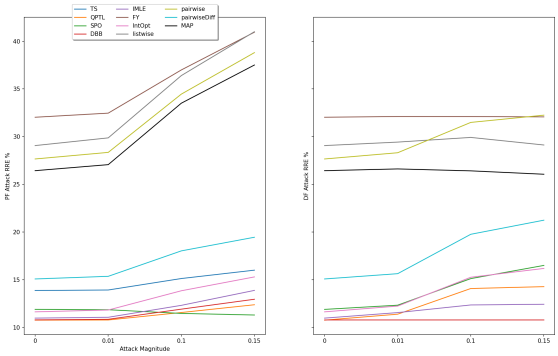

Figure 8: Knapsack capacity 60, RRE

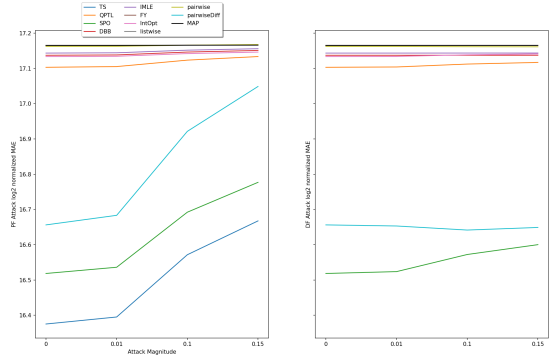

Figure 9: Knapsack capacity 60, MAE

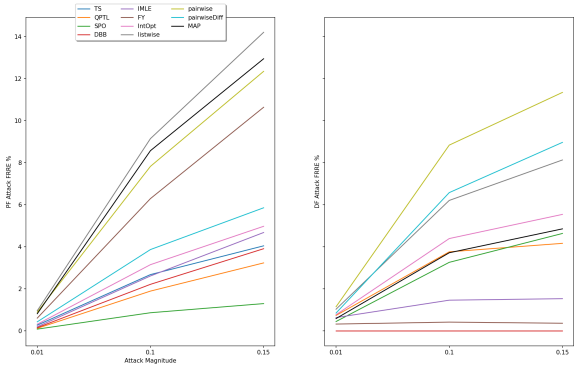

Figure 10: Knapsack capacity 60, FRRE

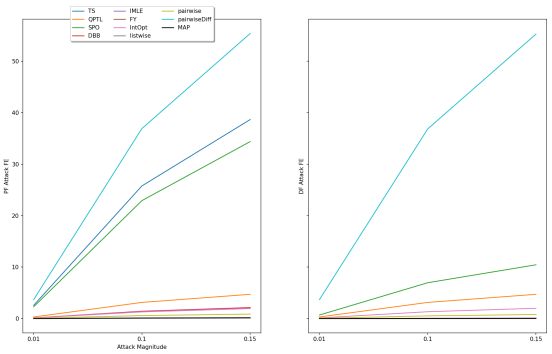

Figure 11: Knapsack capacity 60, FE

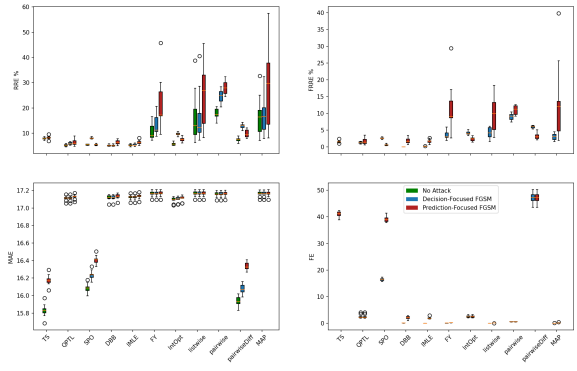

Figure 12: Knapsack capacity 120, Perturbation Magnitude 0.15

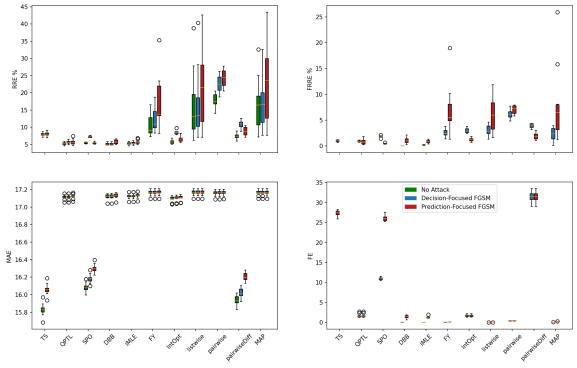

Figure 13: Knapsack capacity 120, Perturbation Magnitude 0.1

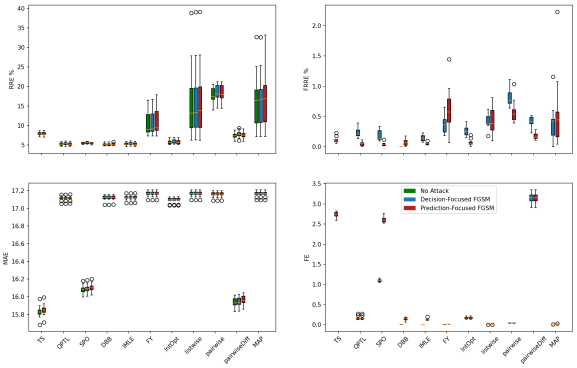

Figure 14: Knapsack capacity 120, Perturbation Magnitude 0.01

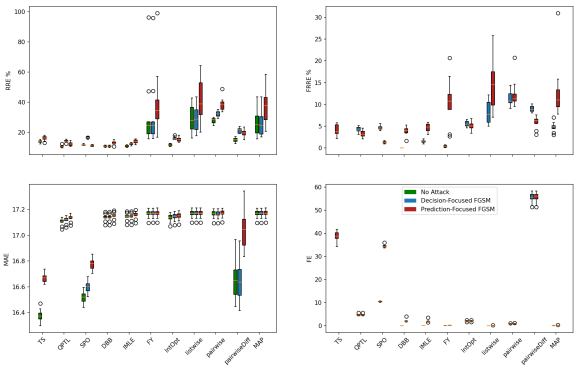

Figure 15: Knapsack capacity 60, Perturbation Magnitude 0.15

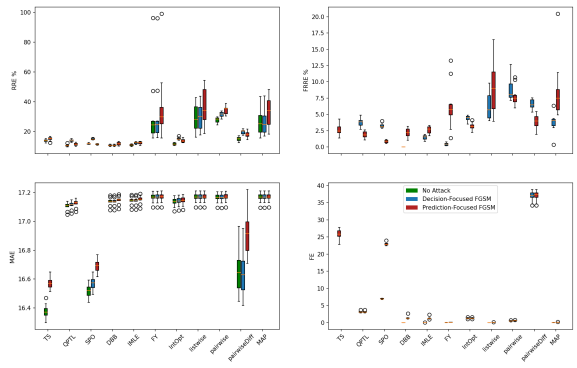

Figure 16: Knapsack capacity 60, Perturbation Magnitude 0.1

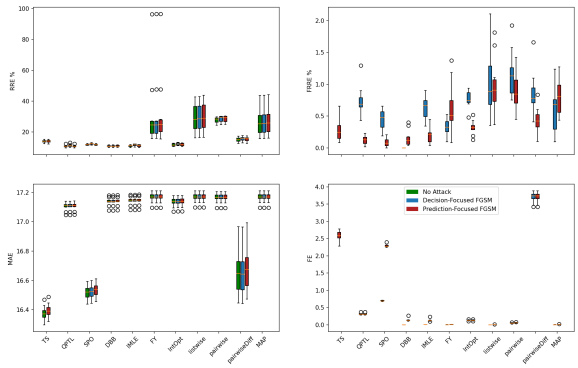

Figure 17: Knapsack capacity 60, Perturbation Magnitude 0.01

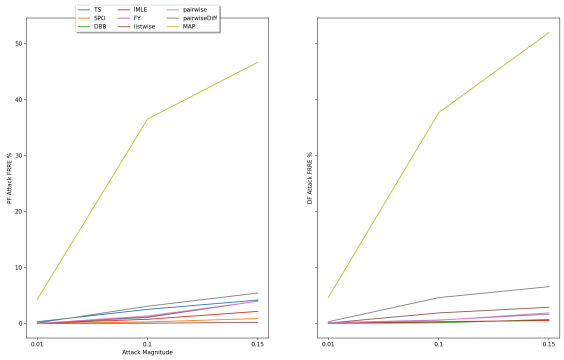

Figure 20: Warcraft img size 12×12, FRRE

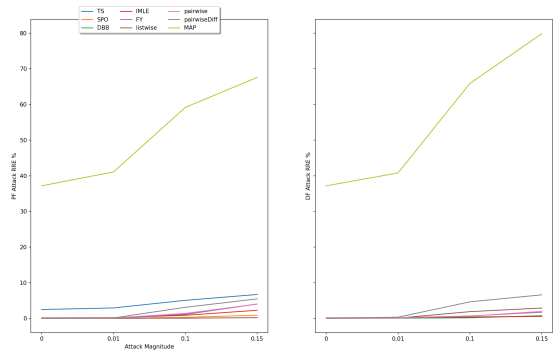

Figure 18: Warcraft img size 12×12, RRE

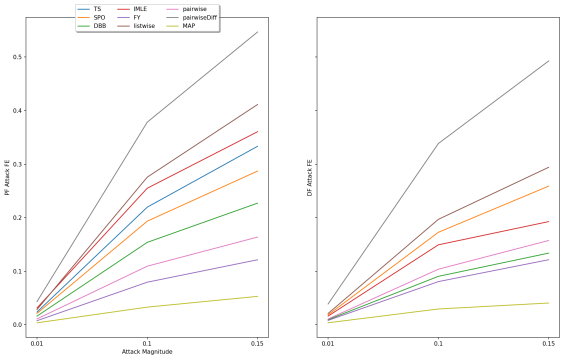

Figure 21: Warcraft img size 12×12, FE

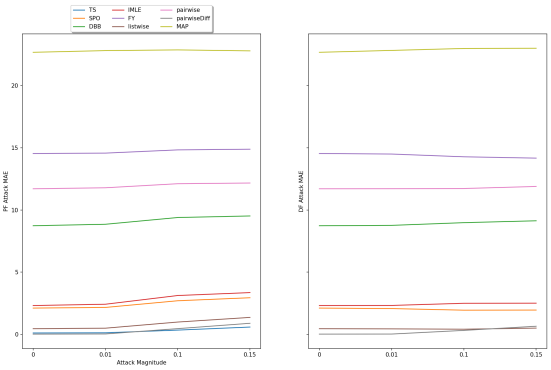

Figure 19: Warcraft img size 12×12, MAE

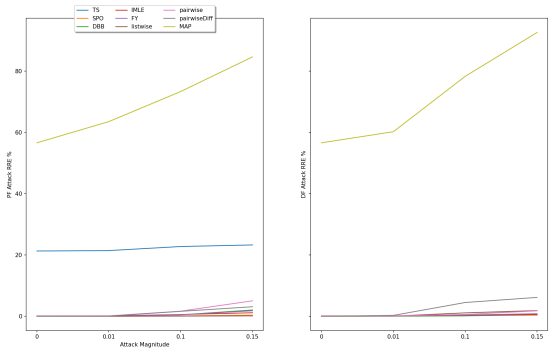

Figure 22: Warcraft img size 24×24, RRE

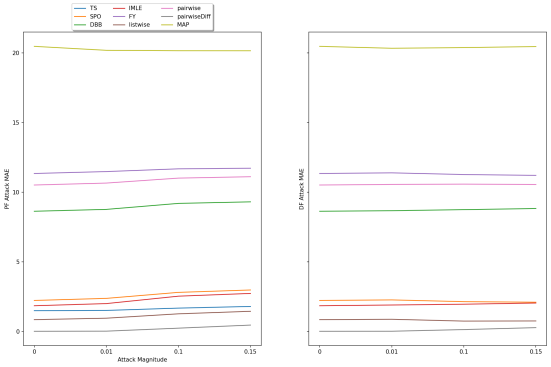

Figure 23: Warcraft img size 24×24, MAE

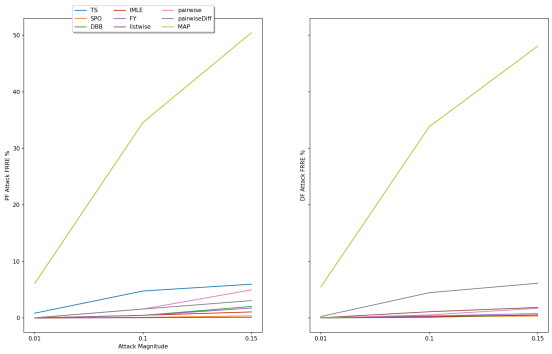

Figure 24: Warcraft img size 24×24, FRRE

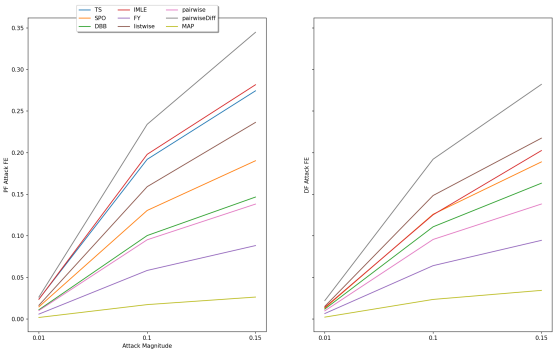

Figure 25: Warcraft img size 24×24, FE

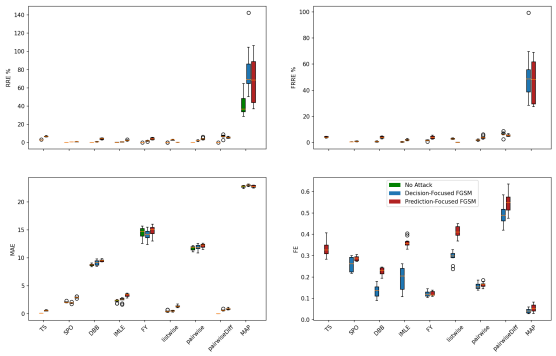

Figure 26: WarCraft img size 12×12, Perturbation Magnitude 0.15

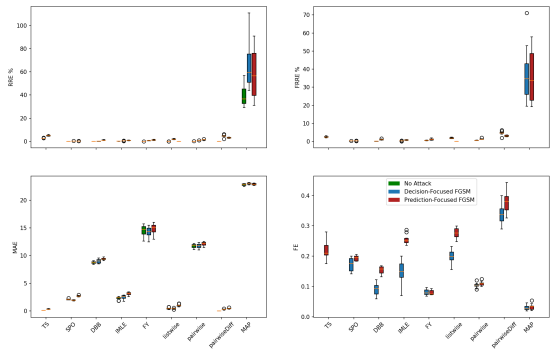

Figure 27: WarCraft img size 12×12, Perturbation Magnitude 0.1

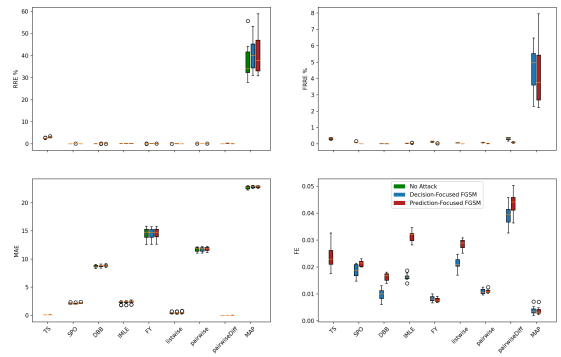

Figure 28: WarCraft img size 12×12, Perturbation Magnitude 0.01

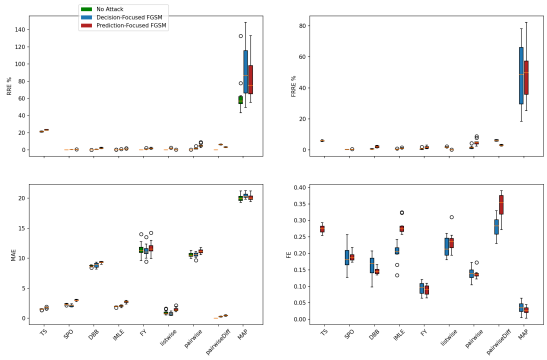

Figure 29: WarCraft img size 24×24, Perturbation Magnitude 0.15

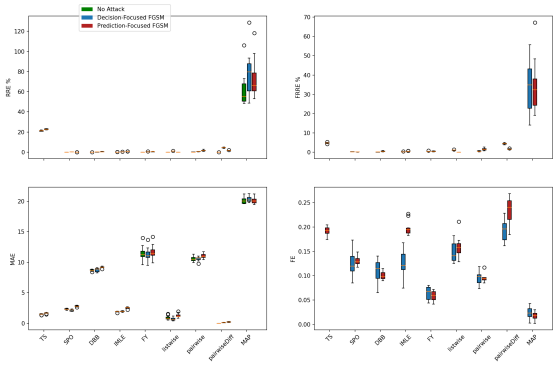

Figure 30: WarCraft img size 24×24, Perturbation Magnitude 0.1

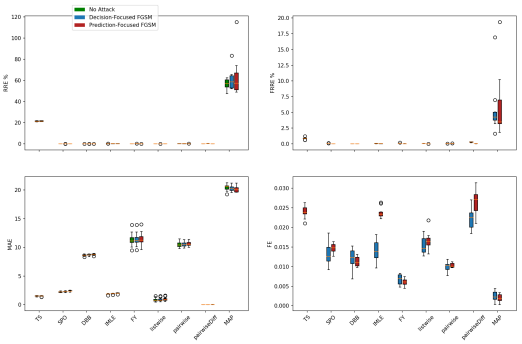

Figure 31: WarCraft img size 24×24, Perturbation Magnitude 0.01

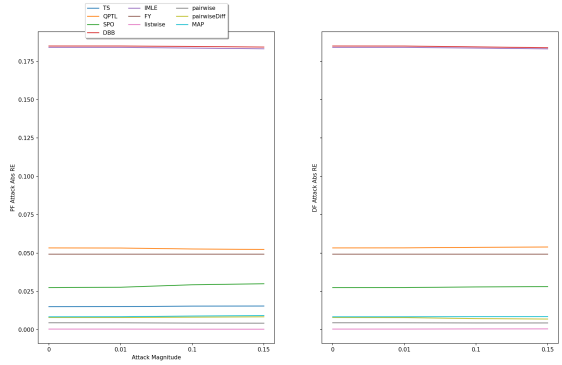

Figure 32: Portfolio deg 16, Absolute RE

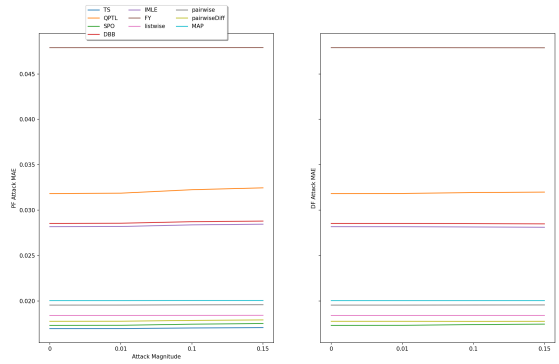

Figure 33: Portfolio deg 16, MAE

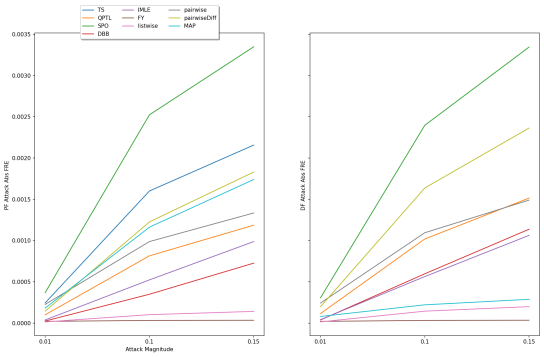

Figure 34: Portfolio deg 16, Absolute FRE

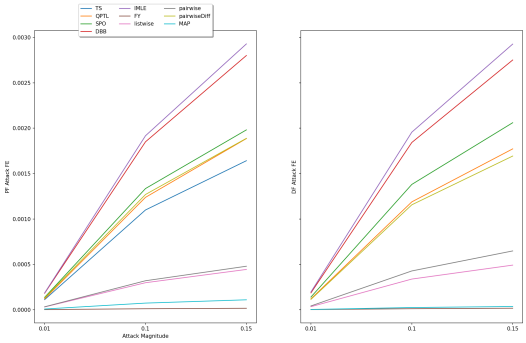

Figure 35: Portfolio deg 16, FE

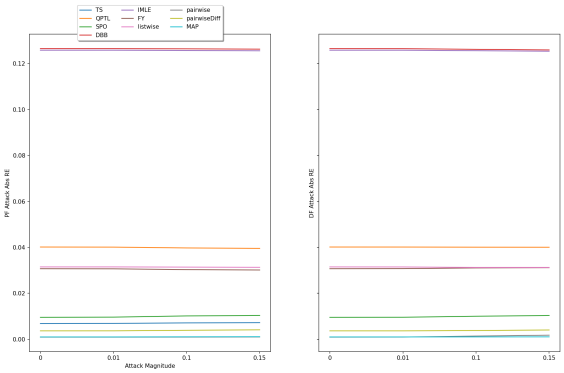

Figure 36: Portfolio deg 1, Absolute RE

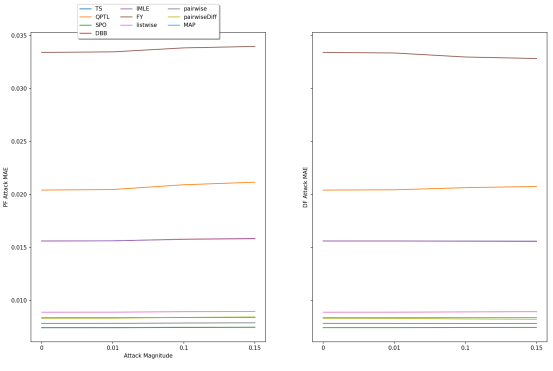

Figure 37: Portfolio deg 1, MAE

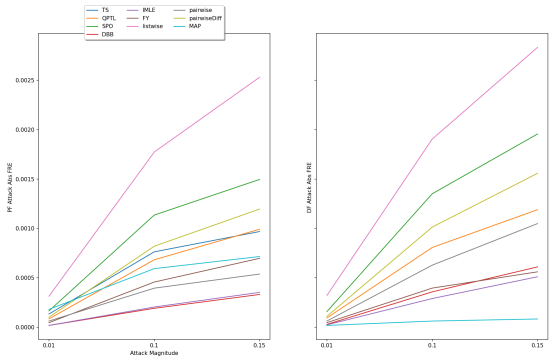

Figure 38: Portfolio deg 1, Absolute FRE

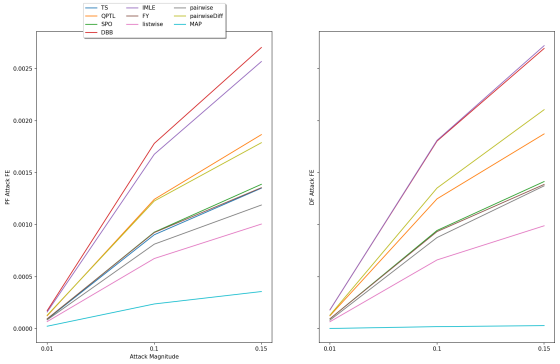

Figure 39: Portfolio deg 1, FE

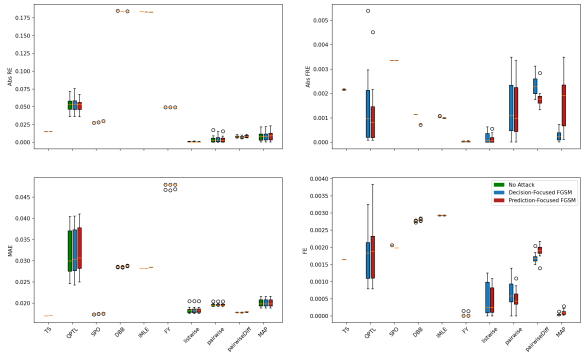

Figure 40: Portfolio Degree 16, Perturbation Magnitude 0.15

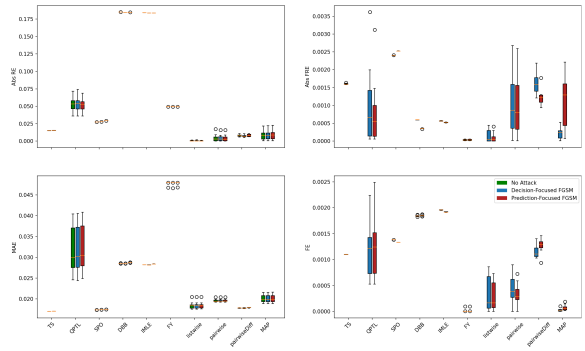

Figure 41: Portfolio Degree 16, Perturbation Magnitude 0.1

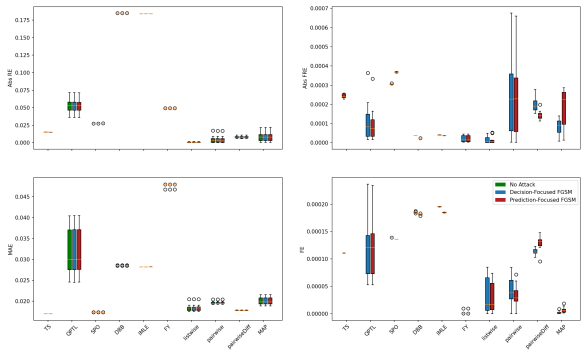

Figure 42: Portfolio Degree 16, Perturbation Magnitude 0.01

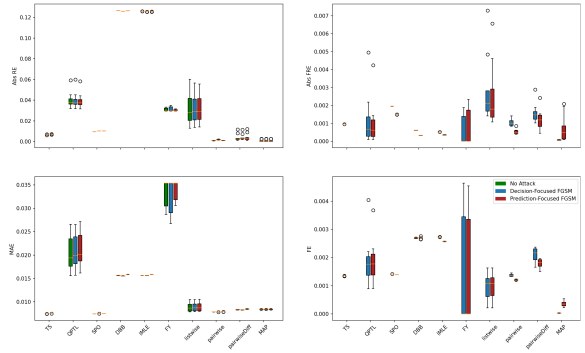

Figure 43: Portfolio Degree 1, Perturbation Magnitude 0.15

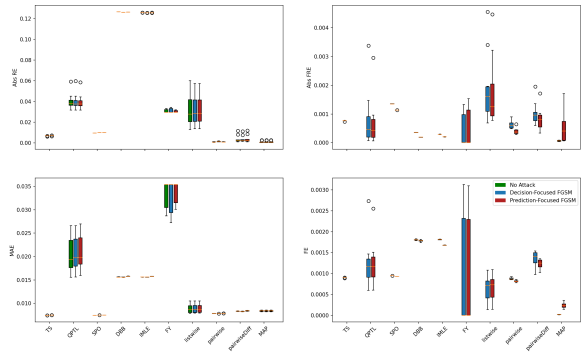

Figure 44: Portfolio Degree 1, Perturbation Magnitude 0.1

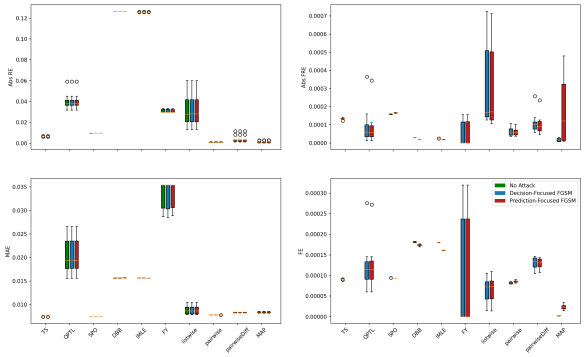

Figure 45: Portfolio Degree 1, Perturbation Magnitude 0.01