# Correctional Regret for Predict+Optimize with Unknown Objectives and Constraints

**Xinyi Hu**[1] , **Jasper C.H. Lee**[2,3] , **Jimmy H.M. Lee**[1] and **Allen Z. Zhong**[1]
[1]Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]Department of Computer Sciences
[3]Institute for Foundations of Data Science
University of Wisconsin–Madison, WI, USA
{xyhu,jlee,zwzhong}@cse.cuhk.edu.hk, jasper.lee@wisc.edu

## Abstract

Predict+Optimize, which combines machine learning and constrained optimization, tackles optimization problems containing parameters that are unknown at the time of solving. Prior works focus on optimization problems with unknown objectives but known constraints, and use the regret function as the loss function. When the constraints are unknown, the problem becomes more complex since the solution solved using estimated parameters may end up being infeasible under the true parameters. Thus, the regret function, which does not consider the solution feasibility, is no longer applicable. We introduce the novel notion of *correctional regret* to account for correcting infeasible estimated solutions before computing the regret. Experimentation shows better performance for our proposal over classical and state of the art approaches.

## 1 Introduction

In the intersection of machine learning and constrained optimization, the Predict+Optimize framework tackles optimization problems with parameters that are unknown at solving time. The task is to i) predict the unknown parameters, then ii) solve the optimization problem using the predicted parameters, such that the resulting solutions are good even under true parameters. Traditionally, the parameter prediction uses standard machine learning techniques, with error measures independent of the optimization problem. Thus, the predicted parameters may in fact lead to a low-quality solution for the (true) optimization problem despite being "high-quality" for the error metric. The Predict+Optimize framework uses the more effective *regret function* [Demirović *et al.*, 2019a; Elmachtoub and Grigas, 2022; Guler *et al.*, 2022] as the error metric, capturing the difference in objective (computed under the true parameters) between the estimated and true optimal solutions. However, the regret function is usually not differentiable, and gradient-based methods do not apply.

Prior works have focused on the regime where the optimization problems contain an unknown objective but known constraints, and proposed ways to overcome the non-differentiability of the regret. They can be roughly divided into two approaches: *approximation* and *exact*. The former tries to compute the (approximate) gradients of (approximations of) the regret function. Elmachtoub *et al.* [2022] propose a differentiable surrogate function for the regret function, while Wilder *et al.* [2019] relax the integral objective in constrained optimization and solve a regularized quadratic programming problem. Mandy and Guns [2020] focus on mixed integer linear programs and propose an interior point based approach. While novel, approximation approaches are not always reliable. *Exact* approaches exploit the structure of optimization problems to train models without computing gradients. Demirović *et al.* [2019b] investigate problems with the ranking property and propose a large neighborhood search method to learn a linear prediction function. Demirović *et al.* [2020] further extend the method to enable Predict+Optimize for problems amenable to tabular dynamic programming (DP), which uses a coordinate descent method to train linear models for prediction. This work is extended to the the Branch & Learn framework [Hu *et al.*, 2022], where problems solvable with a recursive or iterative algorithm can be tackled in Predict+Optimize. Guler et al. [2022] proposes a divide and conquer algorithm, extending the work of Demirović *et al.* [2020] in a different manner so that the algorithm can deal with problems with linear objective function.

Despite the variety of approaches, Predict+Optimize on optimization problems containing both an unknown objective and unknown constraints remains uncovered. The main challenge is that the solution solved using estimated parameters may end up being *infeasible* under the true parameters—an issue inherent with uncertainty in constraints. We therefore focus on applications where, after the true parameters become known, the solved solution can be modified (perhaps in a restricted manner) into a feasible solution. The regret function designed for fixed solution space is not applicable in this situation, and we propose a novel *correctional regret* loss function. In this setting, an infeasible solution is first converted into a feasible estimation (with respect to true parameters), and then the error of prediction is measured by the objective difference between the true optimal solution and the feasible estimated solution. We give case studies on how to define correctional regret on three different combinatorial optimization

problems. Similar to the original regret function, correctional regret is usually non-differentiable. We therefore incorporate the correctional regret into the Branch & Learn framework [Hu *et al.*, 2022], which can tackle unknown parameters in both objectives and constraints. The experiment results on the maximum flow problem (MFP) and the minimum cost vertex cover (MCVC) problem demonstrate the superior solution quality of the proposed method.

## 2 Problem Description

Without loss of generality, an *optimization problem* is to find $x^* = \arg\min_x obj(x)$ s.t. $C(x)$, where $x \in \mathbb{R}^d$ is a vector of decision variables, $obj : \mathbb{R}^d \to \mathbb{R}$ is a function mapping $x$ to a real *objective value* which is to be minimized, and $C$ is a set of constraints over $x$. Thus, $x^*$ is an *optimal solution* and $obj(x^*)$ is the *optimal value*.

A *parameterized optimization problem (Para-OP)* $P(\theta)$ is an extension of the *optimization problem* $P$:

$$x^*(\theta) = \arg\min_x obj(x, \theta) \text{ s.t. } C(x, \theta)$$

where $\theta \in \mathbb{R}^t$ is a vector of parameters. Both the objective and constraints depend on $\theta$. When the parameters are known, a Para-OP is just an optimization problem.

In *Predict+Optimize* [Demirović *et al.*, 2020], the true parameters $\theta \in \mathbb{R}^t$ for a Para-OP are unknown at solving time, and *estimated parameters* $\hat{\theta}$ are used instead. Suppose each parameter is estimated by $m$ features. The estimation will rely on a machine learning model trained over $n$ observations of a training data set $\{(A^1, \theta^1), \ldots, (A^n, \theta^n)\}$, where $A^i \in \mathbb{R}^{t \times m}$ is a *feature matrix* for $\theta^i$, so as to yield a *prediction function* $f : \mathbb{R}^{t \times m} \to \mathbb{R}^t$ for parameters $\hat{\theta} = f(A)$.

The quality of the estimated parameters $\hat{\theta}$ is measured by the *regret function*, which is the objective difference between the *true optimal solution* $x^*(\theta)$ and the *estimated solution* $x^*(\hat{\theta})$ under the true parameters $\theta$. Formally, we define the regret function $Regret(\hat{\theta}, \theta) : \mathbb{R}^t \times \mathbb{R}^t \to \mathbb{R}_{\geq 0}$ to be:

$$Regret(\hat{\theta}, \theta) = obj(x^*(\hat{\theta}), \theta) - obj(x^*(\theta), \theta)$$

where $obj(x^*(\hat{\theta}), \theta)$ is the *estimated optimal value* and $obj(x^*(\theta), \theta)$ is the *true optimal value*. Following the empirical risk minimization principle, Elmachtoub, Liang and McNellis [Elmachtoub *et al.*, 2020] choose the prediction function to be the function $f$ from the set of models $\mathcal{F}$ attaining the smallest average regret over the training data:

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} Regret(f(A^i), \theta^i) \quad (1)$$

When constraints contain unknown parameters, the feasible region is only approximated and the estimated solution may be infeasible when the true parameters are used. Therefore, the regret function, which does not take feasibility into account, becomes inapplicable in this case.

## 3 Correctional Regret

Since the estimated solution may be out of the true solution space, the core idea is to design a *correction* function $f_c$ that maps (a) every feasible solution to itself and (b) each infeasible solution to one in the feasible region. We define *correctional regret* $CReg(\hat{\theta}, \theta) : \mathbb{R}^t \times \mathbb{R}^t \to \mathbb{R}_{\geq 0}$ as:

$$CReg(\hat{\theta}, \theta) = obj(f_c(x^*(\hat{\theta})), \theta) - obj(x^*(\theta), \theta) \quad (2)$$

The definition of $f_c$ is problem and application specific.

Similar to the regret function and following the empirical risk minimization principle, we choose the prediction function to be the function $f$ from the set of models $\mathcal{F}$ attaining the smallest average correctional regret over the training data:

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} CReg(f(A^i), \theta^i) \quad (3)$$

We give case studies on how to define correctional regret on three combinatorial optimization problems as follows.

**0-1 Knapsack** We first demonstrate, using the example of 0-1 knapsack problem with unknown costs, how a correction function can be defined. Consider a project funding problem, which gives a set of projects, each of which needs a cost to conduct and can publish some papers as profit. The aim is to maximize the total profit under a budget, which is an instance of 0-1 knapsack. If the profits for the projects are given but the costs are unknown, we need to consider a case where the projects are selected with the estimated costs, but the total true costs might exceed the available funding.

One trivial correction function is to fund no projects if the total (true) funding required exceeds the budget. A more useful correction function is to remove the adopted projects in the estimated solution one by one in increasing order of profit values until the available funding is sufficient.

**Maximum Flow** The maximum flow problem (MFP) with unknown edge capacities is another example demonstrating what a correction function can be when the unknown parameters are only in constraints. Given a directed graph, where at most one edge exists between any two vertices. Each edge on the graph has a non-negative capacity. The aim is to find the largest possible flow sent from the source $s$ to the terminal $t$ under the constraints that the flow on each edge must be smaller than or equal to the edge capacity. The edge capacities are unknown parameters.

We need to consider a case where the path and flow computed with the estimated capacities might exceed the true capacities of some edges. One correction function is to recompute the blocking flows of the chosen paths with the true capacities, and then augment the paths one by one with the re-computed blocking flows. The ordering of path augmentation is important but computing the best order requires $O(n!)$ time. We can adopt an approximate method: the paths are augmented according to the order of the path augmentation of the Edmonds-Karp algorithm.

**Minimum Cost Vertex Cover** Our last example is a variant of the minimum cost vertex cover (MCVC) problem, where we show how to define a correction function when having unknown parameters in both the objective and constraints. Given a graph $G = (V, E)$, there is an associated *cost* $c \in \mathbb{R}^{|V|}$ denoting the cost of picking each vertex, as

| | Artificial Dataset | | | | Real-life Dataset | | | |
|---|---|---|---|---|---|---|---|---|
| | USANet | | GÉANT | | USANet | | GÉANT | |
| Size | 100 | 300 | 100 | 300 | 100 | 300 | 100 | 300 |
| B&L | 58.6±27.8 | 34.4±16.5 | 22.1±10.7 | 19.4±10.1 | 3.7±3.0 | 3.5±2.7 | 2.3±1.6 | 2.2±1.6 |
| B&L-C | **34.9±18.7** | **33.5±16.7** | **19.2±9.7** | **18.6±9.8** | **2.4±2.3** | **2.6±2.9** | **1.9±1.2** | **1.5±1.4** |
| LR | 36.1±19.4 | 34.2±16.9 | 20.5±9.7 | 19.7±10.6 | 4.4±2.8 | 4.5±2.5 | 2.3±1.5 | 2.6±1.9 |
| $k$-NN | 35.9±17.0 | 34.0±15.6 | 21.0±11.4 | 19.6±10.0 | 5.2±2.6 | 5.7±3.0 | 2.7±1.6 | 3.4±2.0 |
| CART | 43.0±19.1 | 42.8±17.8 | 25.4±15.3 | 24.3±14.9 | 7.7±4.0 | 7.8±3.7 | 4.6±3.2 | 6.2±4.2 |
| RF | 36.6±17.8 | 33.6±15.9 | 20.9±11.6 | 19.3±9.4 | 4.7±2.6 | 5.0±2.7 | 2.6±1.4 | 3.1±1.9 |
| Average TOV | 140.7±38.7 | 137.7±36.7 | 118.2±50.4 | 114.5±49.4 | 81.8±23.0 | 87.1±24.7 | 74.7±23.0 | 77.2±25.0 |

Table 1: Mean correctional regrets and standard deviations for MFP with unknown capacities.

| | Artificial Dataset | | | | Real-life Dataset | | | |
|---|---|---|---|---|---|---|---|---|
| | ABILENE | | PDH | | ABILENE | | PDH | |
| Size | 100 | 300 | 100 | 300 | 100 | 300 | 100 | 300 |
| B&L | 190.6±23.4 | 193.7±15.3 | 140.9±15.1 | 148±12.8 | 16.4±7.2 | 15.3±3.6 | 73.6±15.6 | 73.6±8.5 |
| B&L-C | **186.1±23.3** | **190.6±13.5** | **140.4±16.5** | **146.5±11.3** | **12.2±5.4** | **11.8±2.8** | **54.9±12.3** | **55.9±8.5** |
| LR | 196.1±27.9 | 197.7±14.6 | 149.1±19.5 | 150.1±10 | 16.3±4.8 | 19.3±3.1 | 69.5±12 | 65.2±6.8 |
| $k$-nn | 196.9±27.8 | 198.6±13.4 | 147.3±23.6 | 149.6±11.7 | 32.5±8 | 33.1±4.5 | 74.8±13.3 | 70.5±6.7 |
| CART | 215.5±18.1 | 209.3±13.4 | 153.7±21.2 | 160.8±12.4 | 25.8±9.2 | 28.6±5.7 | 69.9±12.1 | 66±7.4 |
| RF | 199.2±24 | 197.8±15.1 | 148±18.5 | 151±9.7 | 26.4±7.8 | 27.9±4.3 | 69.3±14.6 | 65.3±8 |
| Average TOV | 582.9±24.3 | 579.6±13.6 | 800.6±25.6 | 804.3±14.7 | 272.1±14.4 | 275.3±5.4 | 492.9±27.9 | 491.2±12.8 |

Table 2: Mean correctional regrets and standard deviations for MCVC with unknown costs and edge values.

well as edge values $\ell \in \mathbb{R}^{|E|}$, one real value for each edge. Both the costs and edge values are unknown parameters. The goal is to pick a subset of vertices, minimizing the total cost, subject to the constraint that for all edges *except the one with the smallest edge value*, the edge needs to be covered, namely at least one of the two vertices on the edge needs to be picked. This problem is relevant in applications such as building public facilities. For example, the graph being a road network with edge values being traffic flow, and we wish to build speed cameras at intersections with minimum cost, while covering all the roads except the one with the least traffic.

The edge value estimation might cause an edge to be wrongly removed. The selected vertices might not cover all the edges that need to be covered. If there is an edge not covered by the selected vertices, a correction function is to add both of the edge endpoints to the selection, and incur also the costs of these two vertices.

## 4 Experimental Evaluation

In this section, we present computational results of both artificial and real-life data experiments wherein we empirically examine the quality of the correctional regret function for training prediction models. Following prior works on Predict+Optimize [Demirović *et al.*, 2019a; Elmachtoub and Grigas, 2022], we focus on linear prediction models. We compare the performance of 6 different methods:

1. Branch and learn (B&L) [Hu *et al.*, 2022], an exact Predict+Optimize method, which train the prediction models with the regret function.
2. Branch and learn with correction (B&L-C), an extension of B&L, which train the prediction models with the correctional regret function.
3. 4 classical regression methods: linear regression (LR), $k$-nearest neighbors ($k$-NN), classification and regression tree (CART) and random forest (RF) [Friedman *et al.*, 2001], all of which train the prediction models with their classical loss function.

We conduct experiments on two of the optimization problems mentioned in Section 3: MFP and MCVC, on real-life graphs. For MFP, we use USANet [Lucerna *et al.*, 2009], with 24 vertices and 43 edges, and GÉANT [LLC, 2018], with 40 vertices and 61 edges. Since MCVC is an NP-hard problem, we use two smaller graphs from the Survivable Network Design Library [Orlowski *et al.*, 2007]: ABILENE, with 12 vertices and 15 edges, and PDH, with 11 vertices and 34 edges. In MFP, the edge capacities are unknown parameters. In MCVC, both the costs and edge values are unknown parameters.

We run 30 simulations for each problem configuration. In each simulation, we build datasets consisting of $n \in \{100, 300\}$ pairs of (feature matrix, parameters). In the artificial and real-life datasets, each parameter has 4 and 8 features respectively. The real-life data are from the ICON energy-aware scheduling competition [Simonis *et al.*, 2014] and also appear in previous works on Predict+Optimize (Demirović *et al.* 2019a; 2020). We use 70% of each of the datasets for training and 30% for testing.

Tables 1 and 2 show the mean correctional regrets and standard deviations in the experiment of MFP and MCVC respectively. *Mean correctional regret ± std* is the metric used to demonstrate the performance. At the bottom of the tables, we report also the *average true optimal values* (TOV) to compare the relative error on the artificial and real-life datasets. Note that B&L performs learning with the regular regret but the testing is with the correctional regret, while B&L-C uses correctional regret in both learning and testing.

As shown in Table 1, B&L-C achieves the best performance in all cases. On the artificial dataset, B&L-C obtains 2.79%-40.44% ($n = 100$ and 0.30%-21.73% ($n = 300$) smaller correctional regret in USANet, and 6.34%-24.41% ($n = 100$) and 3.63%-23.46% ($n = 300$) smaller correctional regret in GÉANT than any other methods. The performance differences among different methods are larger in the real-life dataset, and the advantages of B&L-C are more obvious. Contrasting other methods, B&L-C obtains

35.14%-68.83% ($n = 100$) and 25.71%-66.67% ($n = 300$) smaller correctional regret in USANet, and 17.39%-58.70% ($n = 100$) and 31.82%-75.81% ($n = 300$) smaller correctional regret in GÉANT. Besides, we notice that all methods achieve better performance in the real-life dataset. For example, B&L-C achieves roughly 16-24% relative error in the artificial dataset, and roughly 2-3% relative error in the real-life dataset. This is consistent with how the artificial dataset is purposefully designed to be highly non-linear, and thus more difficult to estimate.

Table 2 shows the results for MCVC. B&L-C has the best performance in all cases with the real-life dataset, achieving roughly 4-11% relative error compared to the TOV. Contrasting other methods, B&L-C obtains 25.15%-62.46% ($n = 100$) and 22.88%-64.35% ($n = 300$) smaller correctional regret in ABILENE, and 20.78%-26.60% ($n = 100$) and 14.26%-24.05% ($n = 300$) smaller correctional regret in PDH. On the artificial dataset, all algorithms perform essentially the same, achieving roughly 32% relative error in ABILENE and roughly 18% relative error in PDH.

## 5 Concluding Remarks

We consider the problem of solving optimization problems with unknown parameters in both objective and constraints. Since the regret function does not take solution feasibility into account, to cater to the regime that the solution solved using estimated parameters may be infeasible under the true parameters, we propose a new loss function, called correctional regret. Case studies on how to define correctional regret on 3 different optimization problems: 0-1 knapsack, MFP, and MCVC are provided. Experiment results on MFP and MCVC show better performance for our proposal over classical and state of the art approaches. One bottleneck is that the proposed correctional regret is problem specific and handcrafted. One potential research direction is to explore more automatic ways to create the correctional regret functions for a new optimization problem or define a more general correctional regret to fit with more general problems.

## References


[Demirović *et al.*, 2019a] Emir Demirović, Peter J Stuckey, James Bailey, Jeffrey Chan, Chris Leckie, Kotagiri Ramamohanarao, and Tias Guns. An investigation into prediction+optimisation for the knapsack problem. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 241–257. Springer, 2019.

[Demirović *et al.*, 2019b] Emir Demirović, Peter J Stuckey, James Bailey, Jeffrey Chan, Christopher Leckie, Kotagiri Ramamohanarao, and Tias Guns. Predict+Optimise with ranking objectives: Exhaustively learning linear functions. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 1078–1085, 2019.

[Demirović *et al.*, 2020] Emir Demirović, Peter J Stuckey, Tias Guns, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jeffrey Chan. Dynamic programming for Predict+Optimise. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 1444–1451, 2020.

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

[Elmachtoub *et al.*, 2020] Adam N. Elmachtoub, Jason Cheuk Nam Liang, and Ryan McNellis. Decision trees for decision-making under the predict-then-optimize framework. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2858–2867, 2020.

[Friedman *et al.*, 2001] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, 2001. Volume 1, Number 10.

[Guler *et al.*, 2022] Ali Ugur Guler, Emir Demirović, Jeffrey Chan, James Bailey, Christopher Leckie, and Peter J Stuckey. A divide and conquer algorithm for Predict+Optimize with non-convex problems. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, 2022.

[Hu *et al.*, 2022] Xinyi Hu, Jasper CH Lee, Jimmy HM Lee, and Allen Z Zhong. Branch & learn for recursively and iteratively solvable problems in Predict+Optimize. *arXiv preprint arXiv:2205.01672*, 2022.

[LLC, 2018] MultiMedia LLC. Geant topology map dec2018 copy. https://www.geant.org/Resources/Documents/GEANT_Topology_Map_December_2018.pdf, 2018. Accessed: 2020-09-10.

[Lucerna *et al.*, 2009] Diego Lucerna, Nicola Gatti, Guido Maier, and Achille Pattavina. On the efficiency of a game theoretic approach to sparse regenerator placement in WDM networks. In *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2009.

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

[Orlowski *et al.*, 2007] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference*, April 2007. http://sndlib.zib.de, extended version accepted in Networks, 2009.

[Simonis *et al.*, 2014] Helmut Simonis, Barry O'Sullivan, Deepak Mehta, Barry Hurley, and Milan De Cauwer. Energy-Cost Aware Scheduling/Forecasting Competition, 2014.

[Wilder *et al.*, 2019] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 1658–1665, 2019.