# Differentiable Optimization Layer with an Interior Point Approach *

Jayanta Mandi[0000−0001−8675−8178] and Tias Guns[0000−0002−2156−2155]**

Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels {firstname.lastname}@vub.be

**Abstract.** Combinatorial Optimization has wide applications throughout many industries. However, in many real-life applications, some or all coefficients of the optimization problem are not known at the time of execution. In such applications, those coefficients are estimated using machine learning (ML) models. End-to-end predict-and-optimize approaches which train the ML model taking the optimization task into consideration, have received increasing attention. In case of mixed integer linear program (MILP), previous work suggested adding a quadratic regularizer term after relaxing the MILP and differentiate the KKT conditions to facilitate gradient-based learning. In this work, we propose to differentiate the homogeneous self-dual formulation of the relaxed LP, which contains more number of parameters, instead of the KKT conditions. Moreover, as our formulation contains a log-barrier term, we need not add a quadratic term to make the formulation differentiable.

**Keywords:** data-driven optimization · interior point method · neural networks

## 1 Introduction

A combinatorial optimization problem maximizes (minimizes) an objective function respecting some constraints on the variables where some or all variables can assume values from a discrete set. Applications of combinatorial optimization abounds in operations research literature. However, in many real-life applications some or all coefficients of the optimization problem are not known. For instance, in case of one-day ahead energy-aware scheduling, where certain machines need to be scheduled with the objective of minimizing energy cost, the price of electricity is not known at the time of execution.

To address such a problem, in a predict-and-optimize approach a machine learning (ML) model is used to first estimate those unknown coefficients before solving the combinatorial problem. For instance, Cohen et al. [2] first predicted future demand of products using an ML model and then use the predicted demand to compute the optimal promotion pricing scheme.

One drawback of such a two-stage approach is the ML model fails to incorporate information from the optimization problem. Standard ML models generate

* This doctoral program paper is based on a paper that is currently under review

** Jayanta Mandi is the student and Tias Guns is the supervisor predictions by leveraging historical and contextual data during model training, when the models learn their parameters by minimizing the expected difference between the generated predictions and the groundtruth. In the absence of an interconnection between the ML and the optimization module, the ML model learns its parameters without regard for the downstream optimization problem.

Hence, end-to-end predict-and-optimize approaches, which consider how effective the predicted values are to solve the optimization problem, have received increasing attention. A standard ML model is trained by computing the gradients of the loss. The gradients indicate how much the model parameters need to be changed in order to generate better predictions. The challenge in end-to-end approach lies in passing the gradients to the ML model after computing the loss using the optimization module.

In this work, we consider combinatorial optimization problems that can be formulated as a mixed integer linear program (MILP). MILP has been used, to tackle a number of combinatorial optimization problems, for instance, in organizing bike-sharing system [13], efficient micro-grid scheduling [9], sales promotion planning [2] and more.

Previous work [16] proposed to compute the gradients by differentiating the KKT condition of the relaxed version of the MILP. However, to execute this, we have to add a quadratic regularizer term to the objective. In this work, we instead, propose to differntiate the homogeneous self-dual formulation, which is a common tool in modern interior point optimization techniques. We address numerical challenges of this approach by proposing early stopping of solving the LP. We also investigate whether differentiating the homogeneous self-dual formulation of the relaxed LP, which contains more number of parameters, instead of the KKT conditions, is more effective. Our empirical demonstrates the benefit of differentiating the homogeneous self-dual formulation.

## 2 Problem Description

In the predict-and-optimize setting, the combinatorial problem is intertwined with an ML model. To estimate the unknown parameters of the optimization problem a predictive ML model $f_\phi(\chi)$ consisting of parameters $\phi$ is learned using the training data. For instance, in case of neural network model, the model parameters $\phi$ would be the weights of each layers of the network. On the test instances, first the unknown parameters are estimated using the trained model before solving the optimization problem.

In a standard ML setup, the quality of the predictions is measured by an error-loss metric $\mathbb{L}$. For a regression task, the error is measured by the mean square error, whereas for classification task it is the cross-entropy loss. The learning objective is to minimize the expected value of this error-loss metric.

Neural network ML models learn model parameters $\phi$ by batch gradient descent (and its variants) [14]. In gradient descent (Algorithm 1), the model parameters are updated iteratively by computing the gradients of $\mathbb{L}$ with respect to the parameters.

**Algorithm 1:** Batch gradient descent

```
  Input : training data $\mathcal{D} \equiv \{(\chi_i, c_i)\}_{i=1}^n$
1 Parameter: $\omega$: Learning Rate
2 Initialize $\phi$
3 repeat
4     sample batch $(\chi, c) \sim \mathcal{D}$
5     predict $\hat{c} = f_\phi(\chi)$ ; Compute $\mathbb{L}$ and $\frac{\partial}{\partial \phi}\mathbb{L}$
6     $\phi \leftarrow \phi - \omega * \frac{\partial}{\partial \phi}\mathbb{L}$
7 until convergence;
```

In this work, we consider constraint optmization problem, that can be transformed into the following MILP form:

$$\min_x c^\top x$$
$$\text{subject to } Ax \leq b; x_i \in \mathbb{Z} \text{ for all or some i and } x_i \geq 0 \ \forall i \tag{1}$$

where $x$ is the solution variable and $c, A, b$ are the coefficients of the optimization problem. We denote the optimal solution of Eq.(1) as $x^*(c; A, b)$. Specifically, we consider, the objective coefficient $c$ is unknown and must be estimated from the features $\chi$ using an ML model. In order to train the ML model, we assume, training data $\{(\chi_i, c_i)\}_{i=1}^n$ are available.

**Two-stage learning** One can treat the prediction and the optimization tasks independently. The ML model can be trained to minimize the $E\Big[\mathbb{L}(f_\phi(\chi), c)\Big]$ without considering the optimization module. After training, on the test instances, one can solve the MILP using the predicted coefficients. Such a two-stage approach does not consider minimizing the regret, but minimizes the error between actual and predictions.

This approach might be justified when the ML model generates accurate or near-accurate predictions, but for most real life problems there are errors in predictions. Interestingly, the impact of prediction errors on the optimization problem is not uniform and a two-stage approach fails to take this into account.

## 2.1 End-to-end Training

In predict-and-optimize setup, the predictions are intermediary results. The objective of the learning is to deliver better performance in terms of the optimization problem. Task-loss [3] is a loss function that evaluates the error after the optimization problem using the predictions. For the MILP problem we consider, an immediate choice for the task-loss is the regret, which is the difference in the objective between using predicted $\hat{c}$ instead of apriori unknown c i.e.

$$\text{regret}\Big(\hat{c}, c; A, b\Big) = c^\top (x^*(\hat{c}; A, b) - x^*(c; A, b)) \tag{2}$$

**Algorithm 2:** End-to-end training

```
   Input : $A, b$; training data $\mathcal{D} \equiv \{(\chi_i, c_i)\}_{i=1}^n$
1  Parameter: $\omega$: Learning Rate
2  Initialize $\phi$
3  repeat
4      sample batch $(\chi, c) \sim \mathcal{D}$
5      predict $\hat{c} = f_\phi(\chi)$
6      $x^* \leftarrow$ MILP Predict+Optimize:: Forward Pass          // solution given
        predictions
7      $\frac{\partial x^*}{\partial \hat{c}} \leftarrow$ MILP Predict+Optimize:: Backward Pass
8      $\frac{\partial \text{ regret}}{\partial \phi} = \frac{\partial \text{ regret}}{\partial x^*} \frac{\partial x^*}{\partial \hat{c}} \frac{\partial \hat{c}}{\partial \phi}$
9      $\phi \leftarrow \phi - \omega * \frac{\partial \text{ regret}}{\partial \phi}$
10 until convergence
```

An end-to-end training of the predict-and-optimize problem considers the problem $\min_\phi E\Big[\text{regret}\big(f_\phi(\chi), c; A, b\big)\Big]$. The end-to-end training is much more challenging in this case as the MILP problem is solved for each training instance repeatedly in each epoch. Clearly this makes the learning an expensive process. Moreover, a bigger challenge lies in the nonconvex and discrete nature of the MILP. An ML model would update its parameters in the backward pass, by computing the gradient of the regret with respect to the model parameters (Algorithm 2). Moreover, this gradient is decomposed by chain rule as follows:

$$\frac{\partial \,\text{regret}\big(f_\phi(\chi), c; A, b\big)}{\partial \phi} = \frac{\partial \,\text{regret}\big(f_\phi(\chi), c; A, b\big)}{\partial x^*(\hat{c}; A, b)} \frac{\partial x^*(\hat{c}; A, b)}{\partial \hat{c}(\phi)} \frac{\partial \hat{c}(\phi)}{\partial \phi} \tag{3}$$

From the definition of regret in Eq. (2), we recognize the first term is equal to $c$. The third term is the gradient of the prediction $\hat{c}$ with respect to the model parameters $\phi$ and computed seamlessly by a deep learning architecture [11]. Notice, the second term is the gradient of the solution $x^*(\hat{c})$ with respect to the coefficient $\hat{c}$ i.e. it is a differentiation through the argmin operator. The problem in argmin differentiation for the discrete MILP problem is argmin is a piece-wise constant function of $\hat{c}$. Hence for every point in the solution space the gradient is either 0 or undefined [4].

In the next section, we will propose a differentiable optimization layer which executes the forward and the backward pass in line 6 and 7 of Algorithm 2.

## 3 Interior Point Approach

First, we overcome the discrete nature of the solution by considering the continuous relaxation of the MILP, which renders it to a continuous LP as follows:

$$min_{x:Ax=b} \; c^\top x; x \geq 0 \tag{4}$$

**Algorithm 3:** MILP Predict+Optimize

```
1  Hyperparameters: $\lambda$-cut-off, $\alpha$: damping factor
2  Forward Pass ($c, A, b$)
3      $x, y, t, \tau, \kappa \leftarrow x_0, y_0, t_0, \tau_0, \kappa_0$        // Initialization
4      repeat
5          Compute search-directions $d_x, d_y, d_\tau$ by solving Eq 7
6          update $x, y, t, \tau, \kappa$ such that $x, t, \tau, \kappa \geq 0$
7          $\lambda \leftarrow \frac{x^\top t + \tau \times \kappa}{p+1}$
8      until $\lambda <$ $\lambda$-cut-off
9      $(x, y, t) \leftarrow (x, y, t)/\tau$
10     return x
11 Backward Pass ()
12     retrieve $(x, y, t; c, A, b)$
13     compute $M = AT^{-1}XA^\top$ to solve Eq (8)
14     $\bar{M} = M + \alpha I$    (Tikhonov damping)
15     return $\frac{\partial x}{\partial c}$ by solving Eq (8) using $\bar{M}$
```

As is well-known, such LP formulation is solvable in polynomial time. Interior point method is one of the efficient tools for solving an LP problem. Let $x^*$ be the optimal solution of Eq. (4); our goal is to compute $\partial x^*/\partial c$.

## 3.1 Forward Pass

In the neural network architecture, the forward pass generates the solution of the optimization problem to compute the regret. In essence, the forward pass is responsible for finding the solution to the optimization problem. For this purpose, we have to find an LP solution of Eq. (4) using interior point method. State-of-the-art interior point methods solve Eq. (4) by forming the following homogeneous self-dual (HSD) [18] formulation:

$$\begin{aligned} Ax - b\tau &= 0 \\ A^\top y + t - c\tau &= 0 \\ -c^\top x + b^\top y - \kappa &= 0 \\ t &= \lambda X^{-1} e \\ \kappa &= \frac{\lambda}{\tau} \\ x, t, \tau, \kappa &\geq 0 \end{aligned} \tag{5}$$

where $\lambda = (x^\top t + \tau \times \kappa)/(p+1)$. The advantage of HSD formulation is it always has a solution, with $(x, \tau)$ and $(t, \kappa)$ being strictly complementary. Moreover, $\kappa/\tau$, in this case, is equal to the duality gap. So non-zero values of $\kappa$ at the solution is used to certify infeasibility [10].

The LP solving starts from an initial value $(x_0, y_0, t_0, \tau_0, \kappa_0)$ of the variables and then they are updated iteratively by respecting the constraints $x, t, \tau, \kappa \geq 0$

[1]. This update is performed for a decreasing sequence of $\lambda$ by solving the following Newton equation system:

$$\begin{bmatrix} A & 0 & 0 & -b & 0 \\ 0 & A^\top & I & -c & 0 \\ -c^\top & b^\top & 0 & 0 & -1 \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & \kappa & \tau \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_t \\ d_\tau \\ d_\kappa \end{bmatrix} = \begin{bmatrix} \hat{r}_p \\ \hat{r}_d \\ \hat{r}_g \\ \hat{r}_{xt} \\ \hat{r}_{\tau\kappa} \end{bmatrix} = - \begin{bmatrix} \eta(Ax - b\tau) \\ \eta(A^\top y + t - c\tau) \\ \eta(-c^\top x + b^\top y - \kappa) \\ Xt - \gamma\lambda e \\ \tau\kappa - \gamma\lambda \end{bmatrix} \tag{6}$$

Here, $(d_x, d_y, d_t, d_\tau, d_\kappa)$ are the updates of the corresponding variables $(x, y, t, \tau, \kappa)$. Moreover, rewriting the last two inequalities as $d_t = X^{-1}(\hat{r}_{xt} - Td_x)$ and $d_\kappa = (\hat{r}_{\tau\kappa} - \kappa d_\tau)/\tau$, we can write Eq. 6 in more compact form:

$$\begin{bmatrix} -X^{-1}T & A^\top & -c \\ A & 0 & -b \\ -c^\top & b^\top & \kappa/\tau \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_\tau \end{bmatrix} = \begin{bmatrix} \hat{r}_d - X^{-1}\hat{r}_{xt} \\ \hat{r}_p \\ \hat{r}_g + (\hat{r}_{\tau\kappa}/\tau) \end{bmatrix} \tag{7}$$

Solution of Eq. (7) is well-documented in LP literature. Readers can refer to [1] for details about the solution process.

## 3.2 Gradient Computation by homogeneous self-dual formulation

To compute the gradient $\partial x/\partial c$ for the backward pass, we propose to differentiate the HSD formulation in Eq. (5). Differentiating Eq. (5) with respect to $c$ and making use of $t = \lambda X^{-1}e$, we write the following matrix equation:

$$\begin{bmatrix} -X^{-1}T & A^\top & -c \\ A & 0 & -b \\ -c^\top & b^\top & \kappa/\tau \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial c} \\ \frac{\partial y}{\partial c} \\ \frac{\partial \tau}{\partial c} \end{bmatrix} = \begin{bmatrix} \tau I \\ 0 \\ x^\top \end{bmatrix} \tag{8}$$

We draw the attention of the readers to the similarity between Eq. (7) and Eq. (8). Eq. (7) is solved for updating the variables in the forward pass, whereas Eq. (8) is solved for computing the gradient in the backward pass. As Eq. (7) and Eq. (8) differs only in right hand side, we solve both in the same way.

## 3.3 Implementation consideration

***Early stopping*** While solving Eq. (7) or Eq. (8) we have to solve a matrix equation of the form $Mv = r$, where $M = AT^{-1}XA^\top$. To obtain a solution $M$ should be a Positive Definite (PD) matrix.

As mentioned in section 3.1, at the optimal $(x, \tau)$ and $(t, \kappa)$ are strictly complementary, which implies $\lambda = (x^\top t + \tau \times \kappa)/(p+1)$ should be 0. This suggests as we approach the optimal point, $\lambda$ becomes close to zero. So effectively at the optimal point $\lambda \to 0 \implies X^{-1}T = \lambda X^{-2} \to 0$. In such situation we cannot solve Eq. (8) because of $M$ not being a PD matrix.

In order to circumvent this, we propose to stop the forward pass solver as soon as $\lambda$ becomes smaller than a given $\lambda$-cut-off; that is before $\lambda$ becomes too small for a numerical stable solution.

***Tikhonov damping*** Even after this fix, we found $M$ not to be PD for some instances. To overcome this, we replace $M$ by its Tikhonov damped [8] form $\bar{M} := M + \alpha I$; $\alpha > 0$ being the damping factor. We use an $\alpha$ value of $10^{-3}$.

## 4 Experimental Results

In this section we evaluate our proposed approach versus the state of the art.

***Experimental Setup*** The optimization problem is a resource-constrained day-ahead job scheduling problem to minimize total energy cost (see [15]). Tasks must be assigned to a given number of machines, where each task has a duration, an earliest start, a latest end, resource requirement and a power usage, and each machine has a resource capacity constraint. Also, tasks cannot be interrupted once started, nor migrated to another machine and must be completed before midnight. Time is discretized in 48 timeslots. As day-ahead energy prices are not known, they must be predicted for each timeslot first using an ML model. The energy price data is taken from the Irish Single Electricity Market Operator (SEMO) [7]. It consists of historical energy price data at 30-minute intervals from 2011-2013. Out of the available 789 days, 552 (70%) are used for training, 60 (8%) for validation and 177 (22%) for testing. Each timeslot instance has calendar attributes; day-ahead estimates of weather characteristics; SEMO day-ahead forecasted energy-load, wind-energy production and prices; and actual wind-speed, temperature, $CO_2$ intensity and price. Of the actual attributes, we keep only the actual price, and use it as a target for prediction.

| | Two-stage | QPTL | SPO | HSD Interior Point |
|---|---|---|---|---|
| MSE-loss | **740** **(8)** | 5106 (37) | 4094 (318) | 3370 (360) |
| Regret | 16290 (1310) | 15398 (2725) | 13081 (105) | **12945** **(1442)** |

**Table 1.** Comparison among approaches for the Energy Scheduling problems Average and standard deviation of MSE-loss and regret on test data

For prediction, we use a feedforward linear neural network. The learning rate, epochs and weight decay are considered as hyperparameters and they are selected by grid search on the validation set. A damping factor value of 0.001 was used for resolving the numerical warnings due to matrix not being PD. The neural network and the MILP model have been implemented using PyTorch 1.5.0 [12] and Gurobipy 9.0 [6], respectively. As mentioned before, we use continuous relaxation of the MILP to train the neural network model. But we are going report the discrete regret on test dataset. We evaluate the regret based on discrete problem because we train the ML model to do well with respect to the discrete MILP problem.

We compare the HSD-interior point method with a two-stage approach, the *QPTL* (quadratic programming task loss) [17] approach and the SPO approach [5] in Table 1, where we report the average and standard deviation (in the brackets) of 10 runs of the prediction-losses and the regrets of each approach. It reveals that our proposed HSD-interior point approach performs marginally better than state-of-the-art approaches.

## 5 Conclusion and Future Directions

We presented a novel framework for end-to-end training of predict-and-optimize class of problems. Specifically we derived an algorithmic formulation, based on an interior point method to differentiate an LP problem, which is deployable in end-to-end neural network architectures. In so doing, we opened up a methodology to which neural networks can be applied without adding an artificial quadratic term. We empirically demonstrate that our proposed approach can outperform the standard two-stage approach as well as the state of the art.

End-to-end neural network models using gradient-based techniques have become popular choice for solving many real-life applications. We believe our proposed approach provides an exciting step towards integrating combinatorial optimization and neural network. We hope this work might hold interest for CP research community as such problems are increasingly being dealt in this community.

Our future work will possibly explore the ideas for improving the equation system solving, as well as for caching and reusing results across the epochs, paving the way for a time-efficient deployment. Another important direction for future work includes considering the effect of continuous relaxation instead of the original discrete combinatorial problem and investigating other continuous surrogate of the discrete problem.

## References

1. Andersen, E.D., Andersen, K.D.: The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. High Performance Optimization pp. 197–232 (2000). https://doi.org/10.1007/978-1-4757-3216-0_8
2. Cohen, M.C., Leung, N.H.Z., Panchamgam, K., Perakis, G., Smith, A.: The impact of linear optimization on promotion planning. Operations Research **65**(2), 446–468 (2017)
3. Demirović, E., Stuckey, P.J., Bailey, J., Chan, J., Leckie, C., Ramamohanarao, K., Guns, T.: An investigation into prediction+ optimisation for the knapsack problem. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 241–257. Springer (2019)
4. Demirović, E., Stuckey, P.J., Bailey, J., Chan, J., Leckie, C., Ramamohanarao, K., Guns, T.: Dynamic programming for predict+ optimise (2020)

5. Elmachtoub, A.N., Grigas, P.: Smart" predict, then optimize". arXiv preprint arXiv:1710.08005 (2017)
6. Gurobi Optimization, L.: Gurobi optimizer reference manual (2020), http://www.gurobi.com
7. Ifrim, G., O'Sullivan, B., Simonis, H.: Properties of energy-price forecasts for scheduling. In: International Conference on Principles and Practice of Constraint Programming. pp. 957–972. Springer (2012)
8. Martens, J., Sutskever, I.: Training deep and recurrent networks with hessian-free optimization. In: Neural networks: Tricks of the trade, pp. 479–535. Springer (2012)
9. Morais, H., Kádár, P., Faria, P., Vale, Z.A., Khodr, H.: Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming. Renewable Energy **35**(1), 151–156 (2010)
10. O'donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. Journal of Optimization Theory and Applications **169**(3), 1042–1068 (2016)
11. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
13. Raviv, T., Tzur, M., Forma, I.A.: Static repositioning in a bike-sharing system: models and solution approaches. EURO Journal on Transportation and Logistics **2**(3), 187–229 (2013)
14. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)
15. Simonis, H., O'Sullivan, B., Mehta, D., Hurley, B., Cauwer, M.D.: CSPLib problem 059: Energy-cost aware scheduling. http://www.csplib.org/Problems/prob059
16. Wilder, B., Dilkina, B., Tambe, M.: Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1658–1665 (2019)
17. Wilder, B., Dilkina, B., Tambe, M.: Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1658–1665 (2019)
18. Wright, S.J.: Primal-dual interior-point methods, vol. 54. Siam (1997)