Université de Montréal

An Exploratory Study of Decision-Focused Learning for Mutli-Commodity Network Design in Transportation

par
Wisang Sugiarta

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en computer science

08, 2023

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

An Exploratory Study of Decision-Focused Learning for Mutli-Commodity Network Design in Transportation

présenté par:

Wisang Sugiarta

a été évalué par un jury composé des personnes suivantes:

Fabian Bastin,	président-rapporteur
Emma Frejinger,	directeur de recherche
Pierre-Luc Bacon,	membre du jury

Mémoire accepté le: . . . .

RÉSUMÉ

Cette thèse présente une exploration du thème de l'apprentissage centré sur la décision (DFL) pour la conception de réseaux. L'approche représente une nouvelle solution combinant l'apprentissage automatique et la programmation mathématique. Ces dernières années, les progrès de l'apprentissage axé sur les décisions ont été couronnés de succès dans des problèmes d'optimisation clés tels que le problème du sac à dos. Cependant, la plupart des recherches existantes traitent des structures d'optimisation où des paramètres inconnus se trouvent dans la fonction objectif. La principale contribution de cette étude est d'appliquer un cadre DFL à un problème d'optimisation où des paramètres inconnus se trouvent dans les contraintes. Plus précisement, l'objectif est d'appliquer une approche DFL au problème *Multi-Commodity Network Design* (MCND).

L'étude se concentre sur des fonctions de perte spéciales telles que *Smart Predict then Optimize* (SPO) et évalue comment un modèle fait des prédictions tout en tenant compte de l'optimisation en aval où les variables cibles sont dans les contraintes.

Nous démontrons les performances de ce cadre par le biais d'expérimentations et de modifications informatiques. SPO a tendance à sous-estimer artificiellement la demande, de sorte qu'il conçoit des réseaux qui augmentent les coûts d'exploitation en raison de sa dépendance à l'égard des pénalités de débordement pour les demandes réalisées. La sous-estimation de la demande est due à la fonction de perte SPO qui se concentre sur la diminution du regret plutôt que sur les erreurs de prédiction dans ce contexte. Les résultats montrent clairement que SPO est capable d'apprendre à réduire le coût des réseaux en prédisant une faible demande, mais qu'il ne représente pas fidèlement la demande réelle.

En conclusion, cette thèse explore l'utilisation du DFL dans le contexte de SPO pour MCND. Notre recherche met en évidence les développements potentiels que la DFL pourrait avoir pour des problèmes de conception de réseau avec un exemple illustratif tout en testant l'une des approches DFL populaires, SPO. Bien que l'approche que nous avons utilisée n'ait pas été en mesure de montrer des résultats très prometteurs, elle suggère que le DFL a du potentiel dans ces types de problèmes si les approches peuvent être mieux adaptés.

mots clés: Apprentissage axé sur la décision, conception de réseaux, optimisation, apprentissage statistique.

ABSTRACT

This thesis presents an exploration into the topic of decision-focused learning (DFL) for network design. The approach represents a novel experiment combining machine learning (ML) with mathematical optimization. In recent years, the progress of DFL has demonstrated success in key optimization problems such as the Knapsack problem. However, the vast amount of existing research deals with optimization structures where unknown parameters are found in the objective function. The primary contribution of this study is to apply a DFL framework to an optimization problem where unknown parameters are found in the constraints. The aim is to apply a DFL approach to the multi-commodity network design (MCND) problem.

We aim to explore the ability of DFL to predict the demand for commodities in the MCND. The study focuses on special loss functions such as Smart Predict then Optimize (SPO) and evaluates how a model makes predictions while accounting for the downstream optimization where target variables are in the constraints.

Through computational experimentation and modifications, we demonstrate the performance of the framework. SPO tends to artificially underpredict demand such that it designs networks that increase operation costs due to its reliance on overflow penalties for realized demands. The underprediction of demand stems from the SPO loss function that focuses on decreasing regret more than prediction errors in this setting. It is clear from the results that SPO is able to learn how to reduce the cost of networks low demand predictions but does not accurately represent the true demand.

In conclusion, this thesis explores the use of DFL in the MCND context. Our research highlights potential developments that DFL could have in mathematical optimization with a motivating example while also testing one of the popular DFL approaches, SPO. While the approach we used was not able to show very promising results, it suggests that DFL does have potential in these types of problems if approaches can be better customized.

Keywords: Decision-Focused Learning, Network Design, Optimization, Machine Learning.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MCND	Multi-Commodity Network Design
SND	Service Network Design
ML	Machine Learning
PO	Predict, then Optimize
SPO	Smart Predict, then Optimize
SPO+	Smart Predict, then Optimize Plus
DFL	Decision-Focused Learning
MIP	Mixed Integer Programming
DL	Deep Learning
AR	Autoregressive
OLS	Ordinary Least of Squares
MLE	Maximum Likelihood Estimation

ACKNOWLEDGMENTS

I am very grateful to my advisor, Emma, for her invaluable patience, expertise, understanding, and help throughout these two years. I cannot imagine having gone through this experience with another advisor. I would also like to thank all the other professors and advisors I have worked with during this degree.

I am also grateful to my cohort members, for the exchange of ideas, late-night projects, and support. Thanks to my friends who have all been important in my life.

Lastly, I would like to thank my family. To my parents who worked so hard to give opportunities they never had. To my brother who is always there. To my expanded family, Lucie, Pierre, and Martin, who have welcomed me into their family and introduced me to many new things. I will always remember Pierre's encouragement in the pursuit of academics. Lastly, to Mathilde who has provided encouragement in everything over the last several years.

CHAPTER 1

INTRODUCTION

1.1 Background

Decision science is a multidisciplinary field that examines the processes behind decision-making, aiming to enhance the performance of systems and processes in various contexts. Decision-making problems in computer science are fundamental challenges that arise across various domains, that try to select optimal courses of action based on given criteria and constraints. These problems encompass a wide spectrum of complexity, from relatively straightforward choices in algorithm design to intricate decisions in artificial intelligence and machine learning. Whether it is determining the most efficient path in graph traversal, selecting the best scheduling strategy in operating systems, or optimizing resource allocation in distributed systems, decision science is at the core of computing science. Researchers and practitioners in the field continually seek new and better algorithms, heuristics, and optimization techniques to tackle these problems efficiently, allowing for smarter, more autonomous, and robust models that can make sound decisions in real-world scenarios.

More specifically, DFL is a pivotal area of research that holds immense importance in various domains. At its core, DFL centers on developing machine learning models that can make predictions, taking into account uncertainties, risks, and consequences. By focusing on decision-making rather than mere predictions, this approach empowers models to not only provide accurate outcomes but also justify their choices, enhancing transparency. The research in DFL encompasses reinforcement learning, loss functions, and game theory, among others, aiming to equip models with the ability to optimize actions based on explicit objectives. As AI continues to pervade critical applications such as autonomous vehicles, healthcare diagnostics, and financial investments, the advancement of DFL becomes imperative to ensure innovative and effective AI-driven decision-making in complex real-world scenarios.

Within the realm of decision science, networks can be a useful data structure for representing real-world problems. From representing complex molecular structures and predator-prey environments in nature to complex financial derivatives and solar systems, networks can represent many problems on Earth. However, more familiar network problems include transportation networks. These networks represent the transportation of people, goods, energy, and almost anything else one can imagine. A network is a set of interconnected things, represented by arcs and nodes, where there can be a flow of anything from node to node carried through connected arcs. Network design problems are important in operations research and transportation science due to their complex representation and their applicability to the real world.

While transportation is vital for the economy and society as a whole, the industry is negatively impacting the environment. In Canada, the transportation sector accounts for 24% of all greenhouse gas emissions [6]. Yet, the industry could improve the planning and design of transport systems as congestion and transporting near-empty containers is a common occurrence. The ability to optimize these systems is vital in reducing society's carbon footprint.

We focus on the transportation of commodities in freight networks. Freight transportation plays a role in the large system that responds to trade in the global market. From crude oil transportation to same-day delivery, transportation logistics must be planned to ensure transportation carriers can be profitable and sustainable. However, with the rise of quick delivery and one-click checkout, consumer demand quickly changes over time. A company's failure to accurately predict and plan for this volatile demand can cause an enormous waste of resources and dissatisfied customers.

Since 1919, Canadian National (CN) Railway has offered the transport of commodities from different hubs across Canada. In contrast to last-mile delivery services, this type of freight transport moves large quantities over large distances such as from ports to urban sorting centers. While last-mile delivery services try to optimize paths from sorting centers to hundreds of individuals using thousands of different paths, rail freight transport is restricted to big hubs and limited railways. Yet, the tactical planning of rail and operations of moving commodities by trains is quite difficult. This problem is an application of multi-commodity network design (MCND). The solution must satisfy all demands, offer competitive prices to customers and minimize operating costs. However, given the demand and supply of commodities at hubs, the problem is solvable with mathematical programming techniques [17].

In spite of the fact that the planning of the transportation of goods in a rail network can be solved, it assumes knowledge of demand. Yet, in practice, decision-making must be done well in advance for freight scheduling. This leads to the problem where demand must be forecast well in advance and often exhibit relatively large errors. Small errors in demand forecasts can have significant impacts on the optimization problems.

It is important to highlight that there are a number of different ways to deal with the underlying uncertainty in commodity demand. Stochastic programming deals with the optimization problems where uncertain parameters are associated with known probability distributions. This allows for the optimization of the expected values as distributions are known [23]. Another alternative to deal with uncertainty is robust optimization which is used when uncertain parameters have no known probabilities. Instead, this approach optimizes over uncertainty sets [5]. Additionally, a predictive modeling approach stands as another, possibly complementary, strategy. Predictive models are trained utilizing historical data and relevant features to make accurate predictions of future demand patterns. A widely used approach in practice consists in predict values of unknown parameters ignoring the surrounding uncertainty (i.e., deterministic optimization).

However, predicting commodity demand is very difficult [21, 37]. This problem is usually solved with a two-step approach. In the first step, we model past demand data and make predictions. With the predictions for a certain time, we can then feed the demand into the mixed integer linear program (MILP) that will output the optimized result. This practice is called "Predict, then Optimize"[28]. Nonetheless, this method fails to account for some important interactions [15]. Namely, while the initial prediction models can be trained to minimize prediction error, lowest error and almost any other error metric, it cannot be trained to optimize the downstream MCND problem.

There is a growing literature dealing with changing demand in optimization problems. Most research estimates demand using historical data and use that demand prediction, either a point prediction or statistical distribution, to formulate an optimization problem and solve it [15]. Those solutions fail to recognize the relationship between the transportation services offered and the demand for those services. Thus, we aim to train prediction models with a loss function capturing the quality of the resulting downstream decisions instead of a classic prediction loss.

Thus, we must look for solutions that can make accurate demand predictions but also account for the result of the optimization problem.

[3] introduced a novel method to make predictions while accounting for a downstream linear program's objective. The work originally attempts to predict stock attributes with a modified loss function that took a downstream portfolio optimization problem into account. This method is called end-to-end learning or more recently, DFL. The objective of this paper is to use a DFL framework to solve the MCND with demand prediction problem. In essence, we look to make demand predictions that also optimize the downstream optimization problem.

The main issue resides in differentiating a modified loss function that includes some result of the optimization problem [19]. We are dealing with design variables (to be defined in Section 1.2) that are constrained to be binary, this poses a problem for differentiability. The thesis explores recent works in DFL that aim to address this issue.

1.2 Problem Description

We consider a fixed-charge capacitated multicommodity network design problem. We define $G = (N, A)$ as a directed graph, where N and A are the sets of nodes and arcs of the network, respectively. The set of arcs is defined by the fixed network and represents how all nodes are connected. Each $a_{ij} \in A$ also has a fixed cost, f_{ij}, which is the cost to open that arc in the network, and a variable cost, c^k_{ij}, which is the cost associated to how much a unit of commodity k costs to move through arc (i, j). Next, K defines the set of commodities where each $k \in K$ is a certain type of good and is required to be transported from its origin $o(k) \in N$ to its destination $s(k) \in N$. The design variables are charged with fixing the configuration of the network. In this problem, $y_{ij} \in \{0, 1\}$. If the arc $(i, j) \in A$

is included in the network, y_{ij} equals 1, otherwise 0. The flow variables determine how the commodities are transported from origin to destination. We use $x_{ij}^k \geq 0$, the quantity of $k \in K$ that transits through arc $(i, j) \in A$.

A simple example can be visualized in Figure 1.1 which represents an instance of a network. Each node labeled V_i, would have a demand for each commodity, d_i^k. Each arc has a capacity, fixed cost, and variable cost. This makes up the components of the network that will be used in this exploratory study.

The problem is modeled using a MILP formulation outlined in [17], where $N(i)$ is the set of nodes excluding node i:

$$z(d) = \min_{x \in X y \in Y} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{1.1a}$$

$$\text{s.t.} \sum_{j \in N(i)} x_{ij}^k - \sum_{j \in N(i)} x_{ji}^k = d_i^k, \forall i \in N, k \in K \tag{1.1b}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \forall (i, j) \in A \tag{1.1c}$$

$$x_{ij}^k \geq 0, \forall (i, j) \in A, \forall k \in K \tag{1.1d}$$

$$y_{ij} \in \{0, 1\}, \forall (i, j) \in A. \tag{1.1e}$$

The objective minimizes the sum of all variable costs and fixed costs associated with the network. The set of first constraints (1.1b) is the demand flow which guarantees that all demand is satisfied at every node for every commodity. This is where the machine learning model predicts all demand for the problem. The second set of constraints assures (1.1c) commodity flow does not surpass capacity in any arc. The third set of constraints (1.1d) assures that all flow variables are zero or positive. The last set of constraints (1.1e) guarantees that the design variables are binary.

However, with real-world data, demand may exceed a given tactical plan in which case extra capacity is introduced or capacity may be relaxed. In our work, we use auxiliary arcs to guarantee a feasible solution for any given a demand vector of the right dimension. We define such auxiliary arcs to connect the source and sink for each com-

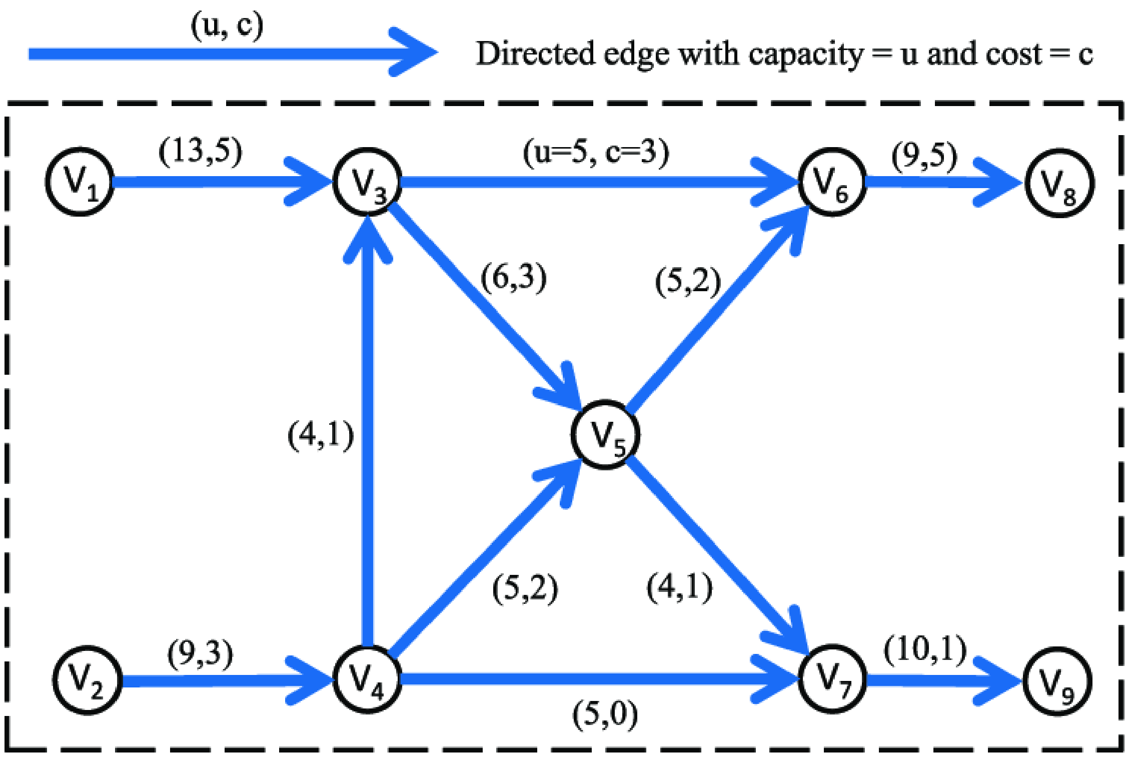


Figure 1.1 – Network Illustration - This figure illustrates a small instance of a network. Arcs have a capacity and cost. Every node has a demand vector where the demand for each commodity is given.

modity with unlimited capacity, zero fixed cost, but very high variable costs. This represents a penalty for not being able to satisfy the demand for a certain commodity and is akin to an overflow penalty.

Furthermore, an important aspect of this MCND problem is demand forecasting. Most recent works mainly deal with predicting demand, d, as a point predictions and then feeding the predictions into the deterministic formulation (1.1a)-(1.1e).

Using a DFL approach to solve an MCND formulation, we want to link the downstream optimization results to the training through the loss function. The key difference in DFL is that the loss function, l_{loss} not only tries to minimize the prediction error but also the regret [3, 26]. A simple regret function example is $l_{regret}(\hat{d}, d) = z^*(d) - z^*(\hat{d})$ which simply measures the difference between the objective value using predicted vector $\hat{d}$ and d. Thus, DFL frameworks include some l_{regret} in its loss function. Yet, this introduces a complication in the ability to differentiate $\frac{dl_{regret}}{dx}$ and $\frac{dl_{regret}}{dy}$. One of the issues is that the decision variables y are constrained to be binary. In this thesis, we explore solutions such that differentiation will be possible.

Specifically, we conduct an exploratory study of using DFL for the MCND problem. The objective is to offer an approach where the model can directly capture the ultimate task-based objective of solving the problem within the context of changing stochastic demand. The main contribution of this study is that it is the first to examine DFL in the context of MCND. While DFL is a growing research topic, they focus on unknown parameters in the objective function and mainly deal with linear programs.

1.3 Contributions and Thesis Structure

The key aspect of this study that we want to highlight is the use of DFL frameworks is not designed to work on the MCND problem. As we will highlight in the next section, DFL frameworks mainly deal with deterministic formulations where the unknown parameters are defined to be in the objective functions. The contribution this study makes is to apply a DFL technique designed for much simpler problems on the MCND and explore results.

In this study, we summarize related works in literature, outline the methodology utilized, analyze exploratory results, conclude the paper and highlight some important future works.

CHAPTER 2

LITERATURE REVIEW

This chapter delves into the research relevant to DFL. We first describe the network design problems, and the MCND in particular. We provide a brief overview of the Predict, then Optimize (SPO) setting, followed by a high-level overview of DFL.

2.1 Network Design

Network design problems have been extensively studied. From solution approaches e.g. exact/heuristic/deterministic/stochastic formulations [17, 23] to forecasting demand using predict, then optimize [28]. Yet, DFL is a newer problem and very little research has been done using its framework applied to network design [15]. Most of this literature review focuses on studies related to the application of DFL in similar problems.

Network design is the optimal selection of arcs so as to minimize the fixed and variable costs of transporting people, information, or commodities through a network [23]. Optimal decisions can be defined as designs that minimize overall costs while satisfying demand. These problems have a combinatorial nature and are NP-Hard. More specifically, [16] introduced the MCND. The original formulation (1.1) from the paper is a MILP. MILPs are defined to have a linear objective function as well as both integer-constrained decision variables and continuous ones [17].

$$z = \min_{w} c^T w \tag{2.1a}$$

$$\text{s.t. } Aw = b \tag{2.1b}$$

$$w \in W \tag{2.1c}$$

Yet, there are some tricks that have been explored to make solving these problems easier. Namely, Lagrangian relaxation (LR) has been shown to greatly decrease the difficulty of some MILPs problems [14, 23]. LR provides the lower bound of a minimization problem. The method can relax one or more constraints such that they are moved to the objective function. For instance,

$$z = \min_{w} c^T w + \lambda^T (Aw - b) \tag{2.2a}$$

$$\text{s.t. } w \in W \tag{2.2b}$$

is the relaxed formulation of Program (2.1a)-(2.1c). Instead of the problem having the hard constraint (2.1b), it can instead be moved to the objective function where it will be optimized.

The nature of this MILP is that it can be solved with commercial software when all parameters are known [22]. However, general-purpose commercial solvers struggle to solve large instances in a reasonable time. Therefore, there are many heuristics and exact methods that have been able to solve these problems. Heuristic-based approaches include simplex-tabu tables and slope-scaling algorithms [9, 11, 17]. Exact methods include branch-and-cut and cutting-plane methods [7, 17, 18].

2.2 Predict, then Optimize

For the MCND, we focus on the unknown parameter of demand from every given instance. It is clear that the MCND is very sensitive to the demand used in the formulation (1.1). Thus, the prediction of demand is vital to the problem. The most common approach is "predict, then optimize".

Previous studies [21, 27] have found that demand forecasting over short- and long-term to be quite difficult for freight and generally. A lot of previous research focuses on auto-regressive time-series models. Autoregressive models are a representation of a random process describing events changing with respect to time [31, 35]. This assumes that the output depends on its past values as well as a stochastic term as:

$$X_t = \sum_{i=0}^{p} \theta_t X_{t-1} + \xi_t. \tag{2.3}$$

It shows that X_t is the result of the product of learned parameters, θ and past values of X, as well as the addition of noise which may be defined as $\xi_t = \mathcal{N}(0, \sigma_t)$. This represents the simplest form of AR models. The most common parameter estimation algorithm is the ordinary least squares (OLS) approach or maximum likelihood estimation (MLE) [35].

2.3 Statistical Learning

Advances in deep learning, Transformers and Long-Short-Term Memory (LSTM) models have shown tremendous improvements in modeling time series and sequential data [20, 24, 36]. The main difference between AR models and ML models is that ML models assume independent and identically distributed data, which is not necessarily true. Yet, these models often show better than baseline performances in forecasting time series data [29, 38, 39].

Supervised ML, in contrast to AR models, relies on n pairs of labeled data, $(x_1, y_1), ..., (x_n, y_n)$, where x_i is a feature vector and y_i is the target label. We try to find a mapping, f_θ, that results in $f_\theta(X) \rightarrow \hat{y}$, where X the set of features and $\hat{y}$ is the predicted target variable. f_θ is defined according to the model (i.e. tree-based models, regressions, etc), where model parameters are defined as θ. We define a scoring function, or loss function, $l(y, \hat{y})$ that expresses the error from the result of the predicted target variable, $\hat{y}$, and the true y, given current parameters.

To estimate parameters θ, we look to the Empirical Risk Minimization (ERM) problem. The ERM problem is possible in the setting of supervised ML. We assume there is a joint probability function, $P(x, y)$ where our training and testing set is drawn (independent and identically distributed from $P(x, y)$ [20]. The risk function, $R(f_\theta)$ associated to a function f_θ, is defined as the expectation of the loss function:

$$R(f_\theta) = \mathbb{E}[l(f_\theta(x), y)] = \int l(f_\theta(x), y) dP(x, y). \tag{2.4}$$

As the risk, $R(f_\theta)$, can not be computed seeing as $P(x,y)$ is unknown, we compute the empirical risk:

$$\frac{1}{n}\sum_{i=1}^{n} l(f_\theta(x_i), y_i). \tag{2.5}$$

Thus, to estimate optimal model parameters given training data, (x_i, y_i), a loss function, $l(y, \hat{y})$, and a prediction model, f_θ, we solve the following ERM problem,

$$\min_\theta \frac{1}{n}\sum_{i=1}^{n} l(f_\theta(x_i), y_i). \tag{2.6}$$

In the realm of optimization, the resolution of this problem is commonly achieved through the application of gradient descent algorithms [20]. One prominent technique in this category is Stochastic Gradient Descent (SGD), which stands as an interactive approach for locating the minima of differentiable functions. This algorithm operates through a sequence of iterative steps, wherein each step involves progressing in the direction opposite to the gradient of the function at the current point, with the ultimate objective of converging towards a minimum value. Throughout its iterative computations over the dataset, the algorithm dynamically updates the parameters based on the following update rule:

$$\theta^{new} = \theta^{old} - \nabla l^{\theta^{new}}. \tag{2.7}$$

Updates occur at every new data point or epoch. The parameters of a model are usually initialized randomly or with a warm start.

In addition, Neural Networks (NN) incorporate some different optimization techniques. They are split into forward and backward propagation. A forward pass consists of calculating all the values at each layer, and all neurons, from the input data, to get $f_\theta = \hat{y}$. A loss function can be calculated from the output values after a forward pass.

On the other hand, backpropagation computes the gradients of a loss function with respect to all the parameters of the network [20]. The goal of backpropagation is to adjust the parameters of the network to reduce prediction error and is considered a type of gradient descent. Starting with the final layers, backpropagation follows the following iterative steps:

1. For each parameter, the gradient is calculated using automatic differentiation, and the parameter is adjusted accordingly, as to follow the approach to the minimum (in the same way as SGD).
2. The next layers can now be calculated using automatic differentiation and parameters adjusted in the same way.

Backpropagation determines how much a specific parameter contributes to the loss function by propagating through a network and calculating gradients with automatic differentiation along the way. It is important to highlight that for both SGD and backpropagation is essential that loss functions are differentiable. This is the crux that DFL research tackles.

2.4 Decision-Focused Learning

Recent developments in ML have led researchers to harness ML to improve downstream optimization problems. A first study [3] found that in the context of a generally similar problem, a portfolio management application, ML and deep learning (DL) models failed to improve profits when trained with a standard loss function. Yet, they were shown to be effective when trained on a final profit-driven loss function. More specifically, when a loss function of the training model accounts for downstream optimization regret function (in this case overall profit/loss), models are able to achieve increased performance. We call this DFL.

Numerous approaches are available for addressing DFL. In the realm of DFL, research predominantly revolves around linear constraint optimization problems, where unknown variables are found linearly in the objective functions. Several methods leverage differentiable optimization layers integrated within NN architectures [1, 2, 25]. These techniques all employ an embedded differentiable optimization layer at various stages of the NN architecture, enabling predictions of unknown parameters to be seamlessly integrated with networks trained via conventional backpropagation. In contrast, alternative approaches introduce innovative loss functions that position optimization as the final stage in the pipeline [4, 13, 32]. These methodologies aim to incorporate loss functions that account for both downstream optimization challenges and prediction errors. In this context, we closely examine two specific methodologies for a more in-depth understanding.

[13] proposes a seminal framework called Smart Predict then Optimize (SPO) that deals with deterministic formulations with unknown parameters in the objective function. This approach works not only with LPs but also any constraint optimization problems where the unknown parameters appear linearly in the objective function. The proposed algorithm introduces specialized loss functions that measure the regret from a downstream optimization problem while also accounting for the prediction errors that occur.

[26, 30] explore applications of DFL to combinatorial optimization problems. They show good results compared to baselines, but all focus on predicting unknown parameters in the objective function, SPO included.

We look more closely at the SPO approach. Given a simple downstream optimization problem:

$$P : z(c) = \min_{w \in W} c^T w \tag{2.8}$$

and a training dataset consisting of pairs $\{(x_1, c_1), .., (x_i, c_i)\}$, the authors propose SPO loss functions $l_{SPO}(c, \hat{c})$ that can account for the optimization problem P while measuring the errors of a prediction model. We denote the predicted cost vector as $\hat{c}$. The SPO loss function is simply: $l^{w^*}_{SPO}(\hat{c}, c) = c^T w^*(\hat{c}) - z^*(c)$, where $w^*(\hat{c})$ are the optimal decision variables given $\hat{c}$. Since ML algorithms attempt to minimize loss, the model would, trivially, incentivize $\hat{c}$ to approach 0. The authors propose a loss function, Smart Predict then Optimize + (SPO+), that minimizes prediction error and reduces objective values,

$$l_{SPO+}(\hat{c}, c) = \max_{w \in W^*(c)} \{c^T w - 2\hat{c}^T w\} + 2\hat{c} w^*(c) - z^*(c), \tag{2.9}$$

where $W^*(\hat{c})$ is the set of optimal w^* given $\hat{c}$. This is called SPO+.

It is proven that the function is convex, and thus differentiable, while also having Fisher Consistency. This means that if the estimator was calculated using a complete population and not only a sample, the true value would be obtained, which is a desirable characteristic of an estimator or loss function [8]. In some cases, a gradient of SPO+ may not exist as there is a min operator. As such, it was proven [13] that there is a subgradient,

$$g(c, \hat{c}) = 2(w^*(c) - w^*(2\hat{c} - c)) \tag{2.10}$$

that can be useful in computationally estimating ML parameters, θ.

There is also literature detailing methods for computing the SPO loss function [13, 30]. The SPO loss function is expensive to compute in its original form, so it is advantageous to have better quick computations. [30] proposes a version of stochastic gradient descent (SGD), using subgradients, to better estimate parameters in the SPO framework, shown in Algorithm 1. The methodology of [30] will be extensively reviewed in Chapter 3.

Algorithm 1 Stochastic Batch gradient descent for the SPO+ loss for regression tasks (batchsize:N) and learning rate α

```
Initialize $\theta$
Sample N training datapoints
for $i$ in 1,...,N do
    predict $\hat{c}_i$ using current $\theta$
    compute subgradient
    $\nabla L_i = w^*(c_i) - w^*(2\hat{c}_i - c_i)$
end for
$\nabla L = \frac{\sum_{i=1}^{N} \nabla L_i}{N}$
$\theta \leftarrow \theta - \alpha * \nabla L * \frac{d\hat{c}_i}{d\theta}$
convergence ($d\theta \leq 0.001$)
```

DFL works differ in one important aspect: where the predictions are in the optimization problem. For instance, [3, 12, 13, 26] all work with a framework where the predictions are located in the objective function, usually predicting costs. Yet, for the MCND problem, the prediction is actually found in the constraints, more specifically the flow constraint as can be seen in the problem description formulation (1.1a)-(1.1e).

The only approach in the literature that is designed for formulations where unknown parameters do not need to be in the objective function is CombOptNet [33]. CombOptNet provides a unique framework where A, B, c in formulation (2.1a)-(2.1c) can all be predicted given a dataset. They suggest incorporating an Integer Linear Problem (ILP) as a differentiable layer in a black box neural network. Essentially, they propose a methodology to find gradients for the mapping $(A,B,c) \to y^*(A,B,c)$. What is unique about this paper is the differentiation of (A,B), unlike the differentiation of c which has been extensively studied [13, 26, 30].

CombOptNet uses the incorporated ILP differentiable layer in a neural network that inputs constraint and objective coefficients and outputs the ILP solution or objective. The ILP layer is just a solver with no modification or relaxation on the solver. The crux is the backward pass on this solver, which means the ILP layer must be differentiable. To estimate parameters in this network, the authors use a descent direction and fixed increments as a proxy to estimate the model's paramters. This leads to promising results in the paper and potential applicability to many DFL problems. However, routinely

This literature review details important works related to the thesis objective. We introduce related works in network design and then MCND. We also introduce the works and notations necessary in ML to understand the Predict, then Optimize (PO) framework. A detailed SPO summary is then shown, highlighting the necessary conditions for its use. CombOptNet is also presented.

Nonetheless, there is an apparent limitation to all previous research with respect to our objective. The DFL literature all deals with formulations where unknown parameters are only in the objective function [34]. Yet, for the MCND and related problems where parameters are in the constraints, to the best of our knowledge no work has been done [34]. In sum, this means that we need to adapt MCND problem such that we can try to apply SPO. We outline the methodology of the experiments in the next section.

CHAPTER 3

METHODOLOGY

This methodology chapter aims to delve into the core principles and techniques underlying DFL for network design. It explores the key components and algorithms involved in our methodology. The goal of our experimental study is to customize the out-of-the-box SPO approach to predict the demand in a network, a problem it is not designed for. We also compare the performance to the baseline (PO) framework. Additionally, we discuss the benefits and potential challenges associated with the application of SPO in network design.

In this chapter, we first present some experimental results for a motivating example to illustrate the potential benefit of DFL for network design. We then briefly discuss why we do not apply CombOptNet (Section 3.2), and in the subsequent sections we delineate the methodology we use in our results (Chapter 4).

3.1 Motivating Example

The main contribution of this work is to explore a DFL framework in the context of the MCND problem. As such, we expect there to be a difference in the error of predictions and the regret. That is to say, given a certain demand point prediction, $\hat{d}_i$, and the true demand, d_i, we expect there to be a difference in the prediction error, $l_{MSE}(d_i, \hat{d}_i)$ (3.2), and the regret error, $l_{regret}(z^*(d_i), z^*(\hat{d}_i))$ (3.1). Notably, we analyze that with changing $\hat{d}_i$, regret and prediction loss are not correlated. If they were, using the downstream regret in the DFL model would not yield any advantage as it would just be some linear combination of the demand prediction loss. The loss function in (2.9) would suggest that this would not serve any useful advantage to a regular demand prediction model, such as PO. However, if the loss and regret are not strongly correlated, this would suggest that the SPO framework could be beneficial.

For this motivating example, we use some of the classical *Canad* network design benchmark instances created by [10]. These instances provide data regarding nodes, arcs, commodities, and a feasible demand for commodities at every node. We also create a smaller instance for quicker computation time. Table 3.I outlines the different Canad instances and one custom instance that we use in the motivating example.

Experiment	# Nodes	# Arcs	# Commodities	Demand Matrix Size	Runtime
Canad #36	20	230	40	20x40	3.1s
Canad #53	30	520	400	5x3	257s
Custom Design	3	6	3	3x3	0.2s

Table 3.I – Network Size and Runtime - This table outlines the sizes and solve time of three different instances.

Firstly, we define regret as in [13], of demand predictions as,

$$l_{regret}(d_i, \hat{d_i}) = z^*(d_i) - z^*(\hat{d_i}). \tag{3.1}$$

We use two additional functions to measure error. We say that y is a vector of predicted variables and y_i is the i^{th} variable in the vector. MSE is defined as,

$$l_{MSE}(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{3.2}$$

.

that we use to quantify errors between vectors.

We also use Log Loss, defined as,

$$l_{LogLoss}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i), \tag{3.3}$$

which quantifies errors in vectors containing only binary values. It is mostly used in evaluating errors in classification predictions but we use it to quantify differences in vectors of design variables as well.

As we further detail below, we run two different simulation experiments to analyze how the model's regret relates to a changing demand vector.

Experiment	Formulation to Change Demand Vector
1	$d_{sim} = a * d, 0.85 \leq a \leq 1$
2	$d_{sim} = d + \mathcal{N}(0, \sigma^2)$

Table 3.II – Formulations for Demand Generation - This table outlines the functions that change an initial demand vector for simulation. Experiment 1 shows a multiplier while Experiment 2 shows the addition of the a random number generated from a normal distribution.

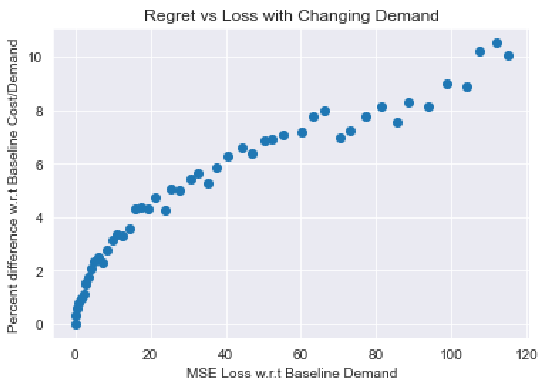
We evaluate the error (or loss) versus the regret of the formulation with respect to a baseline. In this case, we evaluate the log loss of the design variables, which gives intuition of how design variables change with demand predictions. Additionally, we also evaluate the percent difference in objective value with respect to loss in the demand vector. The baseline used is the unique true demand, d, given by the Canad network scenario (Canad only specifies one demand per instance). To simulate demand, we apply a function n times to the demand vector that randomly or systematically changes it. We end up with n d_{sim} vectors that we can compare. Table 3.II outlines the functions that are used for experiments.

This newly generated demand vector, d_{sim}, represents one vector derived from d. We can then calculate both the regrets, $r(d, d_{sim})$ (3.1), and the losses, $l_{MSE}(d_0, d_{sim})$ (3.2). We do 50 simulations and we depict the results in figures that we discuss in the following.

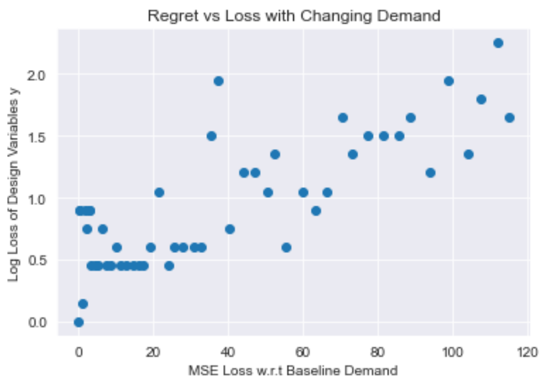
In the first experiment, we apply a function of multiplying the demand by a constant (Experiment 1 in Table 3.II). We ran 50 simulations with linearly increasing constants for each a, where a varies from 0.85 to 1.0.

Figure 3.1a shows regret versus prediction error for Experiment 1. It shows that by linearly increasing the demand vector, the change in percent difference is moderately correlated to the change in MSE of the demand. The expected behavior of this model is that the objective value increases proportionally to the amount of flow on an arc. Yet, if the capacity of the arc is exceeded, we will see non-linear increases in the objective value due to the opening cost of a new arc.

Figure 3.1b shows the difference in design variables versus the prediction loss. We



(a) Objective Regret

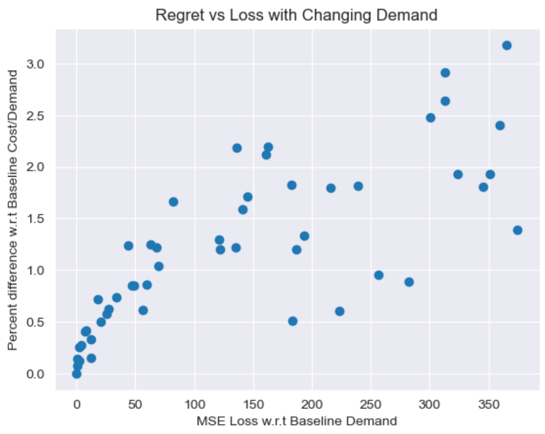


(b) Design Variable

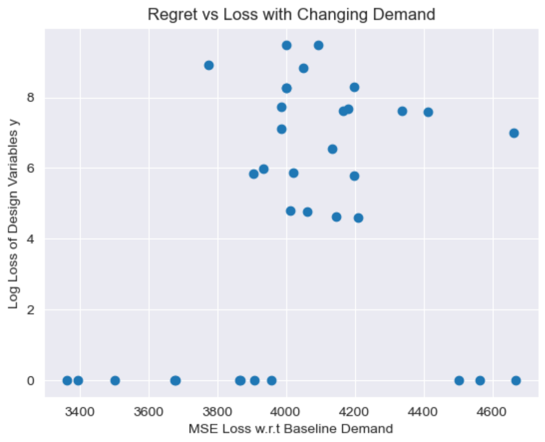
Figure 3.1 – Objective and Design Graphs for Motivating Example - The regret vs. loss of demand simulations for Canad 36, Experiment 2. On the left, the change in regret of the objective function in terms of percent error. On the right, the change in regret in terms of log loss of the design variables chosen.

also see that there is a correlation between the log loss (3.3) of the design variables and MSE of the demand. The log loss measures the difference in the design variable, or arcs, selected for a given demand. As such, we see that the variation between the design variables is quite high for different demands and seems to be unrelated. It is also possible that solutions are not unique and these variations can be due to different optimal solutions. This experiment illustrates that the potential use of DFL could lead to better results. This is due to the fact that there is a correlation between regret and prediction loss, which makes it possible for DFL to learn how to design networks that lead to lower costs.

In the second experiment, a number randomly sampled from a normal distribution is added to the demand vector. In other words, demand can increase or decrease randomly. As such we expect the behavior to deviate from experiment 1. Here, we expect the design variables to change more with different demands and the flow variables to have more variation. Figure 3.2b shows log loss of design variables versus prediction error in Experiment 2. It shows that the variation in design variables is quite high and that the flow variables still increase with respect to the change in demand. Since the demand in this experiment is generated randomly, there is more fluctuation in the selection of design variables, and as a result, different fixed variable costs account for the jumps in the objective value.



(a) Objective Regret



(b) Design Variable

Figure 3.2 – Objective and Design Graphs for Motivating Example - The regret vs. loss of demand simulations for Canad 53 Experiment 2. On the left, the change in regret of the objective function in terms of percent error. On the right, the change in regret in terms of log loss of the design variables chosen.

Figure 3.2a and 3.2b shows the second experiment on the smaller instance. There are only 20 arcs for this instance. In this instance, we strategically set initial values of demand nearing the capacities such that the new randomly generated demands oscillate over and under the capacities. As such, the expected behavior is that there will be large variations in objective values with respect to the demand change. We also see that, to account for the capacities, the design variables fluctuate significantly.

As a result, we see differences in regret and prediction loss especially when capacity is low with respect to the amount of demand. This motivating example hence suggests that SPO could potentially be beneficial

3.2 CombOptNet

After careful analysis and some experimentation with the CombOptNet, we were not able to use it for the MCND problem. There is a difference in formulation where the dimensions of the demand vector must be equal to the dimensions of the decision variables. This stems from the fact that CombOptNet is designed for problems with formulation (2.1a)-(2.1c), where there is an unknown parameter for *each* decision variable. The author's definition of the unknown parameters made is such that its dimensionality must be equal to the dimensionality of the decision variables, w, which in the case of MCND is not true. This is the same problem that is mentioned before in Chapter 2 and later in this chapter, yet with CombOptNet, the modifications needed to make it work for MCND are extensive and deemed outside the scope of this thesis.

3.3 Relaxed MCND Formulation, Prediction Model and Synthetic Data

In this section, we define the relaxed MCND formulation we use in our exploratory study, along with the prediction model and synthetic data.

3.3.1 Relaxed MCND Formulation

All the DFL frameworks outlined in Chapter 2 use deterministic linear programs (LP)s where unknown parameters are in the objective function. The review leads us to believe there are no out-of-the-box algorithms to apply DFL to the MCND problem without heavy modifications. Essentially, SPO can not be applied as is. Instead of modifying SPO, we relax the MCND formulation to explore if this nevertheless leads to predictions that can generate better plans than predictions from PO.

We can relax the flow constraint (1.1b) in Formulation (1.1a)-(1.1e) such that the demand will be in the objective. The following formulation is used in our study:

$$z(d) = \min_{x \in X, y \in Y} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} + \lambda^T \sum_{i \in N} \sum_{k \in K} (d_i^k - \sum_{j \in N(i)} x_{ij}^k - \sum_{j \in N(i)} x_{ji}^k) \tag{3.4a}$$

$$\text{s.t.} \sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \forall (i,j) \in A \tag{3.4b}$$

$$x_{ij}^k \geq 0, \forall (i,j) \in A, \forall k \in K \tag{3.4c}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{3.4d}$$

.

If the penalty λ is sufficiently large, we expect this MILP to approximate the MILP formulation (1.1a)-(1.1e). Recall that a feasible solution always exists as the use of auxiliary arcs ensures there will always be a solution to the MILP.

3.3.2 Prediction Model

We use a simple, easily interpretable linear regression trained with SGD:

$$d = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \tag{3.5}$$

This is a linear regression such that we can use custom loss functions and optimizer to estimate the parameters in Equation (3.5) that minimize the ERM (2.6). Thus, with a given $x_1, \ldots, x_n$ features, and a trained model we can predict the demand.

3.3.3 Dataset

One common assumption in ML is that the data is Independent and Identically Distributed (i.i.d). This is untrue with most demand data [35]. For instance, as demand fluctuates over time it is non-stationary. But, for simplicity, we generate a dataset that follows the i.i.d assumption. First, the data we use in synthetically generated using Sklearn's generative data function. This function outputs generated data by applying a linear regression model with a certain number of informative features. It also adds Gaussian noise with a mean of zero and a variance called noise. We generated 10,000 data points with eight informative features, three noisy features, and nine target variables. The features simply represent principle components. The target variable's dimension matches the cardinality of d. Thus the target variables represent the demand for each commodity at each node (we use the n^{th} node explained in the last section). We add a noise parameter of 1, which represents the standard deviation of the Gaussian noise.

3.4 SPO Implementation Details

In this section, we discuss the details of the implementation of SPO for the MCND. We explain the computations of the n^{th} node, overflow arcs, and decision variables for relaxation and network size.

3.4.1 Sum of Demand Vector Must Equal 0

Due to the demand flow constraint, the demand vector must satisfy an additional constraint, $\sum_{j \in N(i)} x_{ij}^k - \sum_{j \in N(i)} x_{ji}^k = d_i^k$ and $\sum_{i \in N} d_i^k = 0, \forall k \in K$ to ensure feasibility. In sum, the supply and demand for a certain commodity from an origin to a destination must be equal. That is to say, when looking at the demand for a certain commodity k, positive or negative, the sum of all demands, $\sum_{i \in N} \sum_{k \in K} d_i^k$ must be zero or else there would be an excessive of demand that does not exist, making the MILP unsolvable.

No method can be implemented in the prediction algorithm to constrain some predictions based on others so we decide to make no demand prediction on the n^{th}. Instead, we calculate the demand as a source or sink to supply or take demand from the rest of the network. Thus we define the n^{th} node's demand as $d_n^k = -\sum_{i=1}^{n-1} d_i^k, \forall k \in K$. With this n^{th} node definition, we have a demand vector that will always lead to feasible solutions given any demand prediction.

3.4.2 OverFlow Arcs

We want to make sure, given a certain demand, d, that a feasible solution always exists. It is possible to relax the capacity constraint to ensure that the MILP can always be solved, even with demand exceeding network capacity. Though, if we completely relax the constraint, $x_{ij}^k \leq u_{ij} y_{ij}, \forall ij \in A, \forall k \in K$, there is no more enforced relation of x and y. As a concrete example, if the capacity constraint is relaxed and put in the objective [14], then all the terms with y in the objective are: $\sum_{(i,j) \in A} f_{ij} y_{ij} + \sum_{ij \in A} \lambda^T \sum_{k \in K} x_{ij}^k - u_{ij} y_{ij}$. Thus, the incentive is to leave y zero and x_{ij}^k can be non-zero on arcs that are not part of the design. This is undesired behavior and thus we will keep the constraint unrelaxed.

However, it is well known that it is possible to deal with overflow while keeping the capacity constraints [23]. We define extra auxiliary nodes such that $\forall k \in K$ we define an arc from i to j defined as a^{ij-aux} where i is the source and j is the destination for k. We also define flow variables, x^k_{ij-aux}, for the extra auxiliary arcs. However, these arcs, a_{ij-aux} are defined to have unlimited capacity yet very high variable costs. This, in essence, equates to having a very high overflow cost without relaxing any constraint.

3.4.3 Auxiliary Decision Variables to Relax Flow Conservation

After some experimentation and computational review, the use of auxiliary variables for relaxing flow conservation was required for the SPO code to work. This is due to the fact that we want the optimization problem to consider each $d^k_i - \sum_{j \in N(i)} x_{ij} + \sum_{j \in N(i)} x_{ji}, \forall i \in N, \forall k \in K$. However, if we use auxiliary decision variables, $L_{d^k_i} = d^k_i - \sum x_{ij} + \sum x_{ji}$ the solver performs as expected. Thus, the MILP becomes:

$$z(d) = \min \sum_{k \in K} \sum_{(i,j) \in A} c^k_{ij} x^k_{ij} + \sum_{(i,j) \in A} f_{ij} y_{ij} + \lambda^T \sum_{i \in N} \sum_{k \in K} L_{d^k_i} \tag{3.6a}$$

$$\text{s.t.} \sum_{k \in K} x^k_{ij} \leq u_{ij} y_{ij}, \forall (i,j) \in A \tag{3.6b}$$

$$L_{d^k_i} = d^k_i - \sum_{j \in N(i)} x_{ij} + \sum_{j \in N(i)} x_{ji} \tag{3.6c}$$

$$x^k_{ij} \geq 0, \forall (i,j) \in A, \forall k \in K \tag{3.6d}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{3.6e}$$

$$L_{d^k_i} \in \mathbb{R}, \forall i \in N, \forall k \in K \tag{3.6f}$$

Of course, the demand is still considered to be in the objective function, thus SPO+ can be applied.

3.4.4 Network Size

There is a limit on the size of the network we use for SPO+. Table 3.I outlines the run-time for Gurobi Solver to find solutions for the baseline Canad instances. As expected, we see that it increases exponentially as network sizes increase. The utilization of the SPO+ framework requires that for every training point, we solve the MILP with the true demand. Thus, for a dataset with 10,000 data points, before even training the algorithm, we must solve 10,000 MILP instances.

The use of a larger network also requires more training data and features to get accurate results. For all these reasons, we will use the small Custom Design network Table I.I for experimentation.

3.5 Baseline: Predict, then Optimize

As discussed in Section 2, the current approach for the MCND is PO[28]. We want to be able to compare all results to PO so we will use a linear regression model trained with the OLS loss function. Thus, in this approach, the algorithm minimizes demand prediction error with no concern for the downstream optimization problem.

The predicted demand, $\hat{d}$ is then fed directly into the MILP and solved. We can analyze the solved flow variables, x_{ij}^k, design variables, y_{ij}, and objective value, $z^*(\hat{d})$ for a given demand. We now move on to SPO+.

3.6 SPO+ Gradients

To estimate parameters in the predictive model, f_θ, we use the ERM,

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} l_{SPO+}(f_\theta(x_i), d_i), \tag{3.7}$$

The original form of l_{SPO+} 2.9 is expensive to calculate for the gradient descent. Moreover, [13, 30] show that the parameters can be estimated using sub-gradients. The modified stochastic gradient descent, outlined in Algorithm 1 uses sub-gradients to enable model parameters to reach the local minimum of the SPO+ loss function. Since we have two sets of decision variables, x and y, we denote them as w for simplicity. We define the sub-gradients we use to be, (see [13] Proposition 3)

$$\nabla l_{SPO+}(d,\hat{d}) = 2(w^*(d) - w^*(2\hat{d} - d)). \tag{3.8}$$

Using the modified sub-gradients, we can solve the ERM.

Algorithm 1 is used to estimate the parameters of the linear regression algorithm, there is a computational issue, however, where the dimensions of the demand vector must be equal to the dimensions of the decision variables. This stems from the fact that SPO is designed for problems with Formulation (2.1a)-(2.1c), where there is an unknown parameter for *each* decision variable by definition. To estimate parameters, θ, we multiply $\nabla L_i * \frac{\delta \hat{d}_i}{\delta \theta}$. Hence, that means ∇L and $\frac{\delta \hat{d}_i}{\delta \theta}$ must be of the same dimension. However, the dimension of ∇L is the number of decision variables we have, w. The dimension of w is the dimensions of y, which is the number of arcs, $|A|$, in addition to the number of x, which is the number of arcs multiplied by commodities, $|A| * |K|$. So we get the $dim(w) = |A| * |K| + |A|$, $|\nabla L| = |A| * |K| + |A|$.

The dimensionality of $\frac{d\hat{d}_u}{d\theta}$ is simply the dimensionality of d_i^k, so $|\frac{d\hat{d}_u}{d\theta}| = |N| * |K|$. This means that there is a difference of $|A||K| + |A| - |N||K|$. We assume that $|A||K| + |A| - |N||K| \geq 0$. Thus, if we compute this essential gradient, we need $|A||K| + |A| - |N||K|$ more d_i^ks or linear regression output. We define d_{dummy} as extra dummy prediction outputs with dimension $|A||K| + |A| - |N||K|$. We now add d_{dummy} to our target variables in the dataset as $d_{dummy} = [0,..,0]$. The ML model will then output $d_i^k : d_{dummy}$, but only d_i^k will be used in the MCND MILP solver. The dummy outputs are just used in order to compute the gradients and do not impact on either the ML model or MILP.

3.7 DFL Methodology

After showing the necessary modifications to the experimental setting and formally defining the prediction model (3.5), the optimization problem (3.6), and the data, we outline the experimental methodology here.

It is important to remember that the framework of SPO is similar to that of PO, where predictions are made using a predictor and then the output is solved to get the objective value and decision variables. The difference lies in the loss function of the predictor.

We have the optimization problem (3.6), with decision variables w, for which we aim to predict demands that represent the unknown parameters in the objective function. The goal of the exploration is to determine if the objective values of predicted demand using the SPO approach outperform those from a regular loss function. That is to say, will SPO reduce overall objective costs while also keeping predictions accurate.

To do so, we use Algorithm 1 to estimate parameters in the prediction model. The algorithm utilizes the subgradients (2.10) of the SPO+ loss function (2.9) that make predictions based on reducing regret while also keeping accurate demand predictions. Since we try to predict demands, d, we can reexamine the SPO+ loss function:

$$l_{SPO+}(\hat{d},d) = \max_{w \in W^*(d)} \{d^T w - 2\hat{d}^T w\} + 2\hat{d}w^*(d) - z^*(d). \tag{3.9}$$

where the subgradients are found to be:

$$g(c,\hat{c}) = 2(w^*(d) - w^*(2\hat{d} - d)), \tag{3.10}$$

where we denote the two decision variables, x and y as w for simplicity. The particular details of SPO+ have been discussed.

To illustrate the pipeline, we include Figure 3.3. The predictor takes in a feature vector, X, associated with true demand, d. The MCND optimizer will solve for $w^*(d)$ and $z^*(d)$. The predictor outputs $\hat{d}$ which is then be fed into the MCND optimizer to get $w^*(\hat{d})$ and $z^*(\hat{d})$. The SPO loss function then has all parameters needed and Algorithm 1 can then change predictor parameters according to this iteration.

Once trained, the results of the exploration can be studied. We can evaluate how accurate demand predictions are while verifying the regret.

After showing the necessary modifications and methodology, in the next section, we present the results.

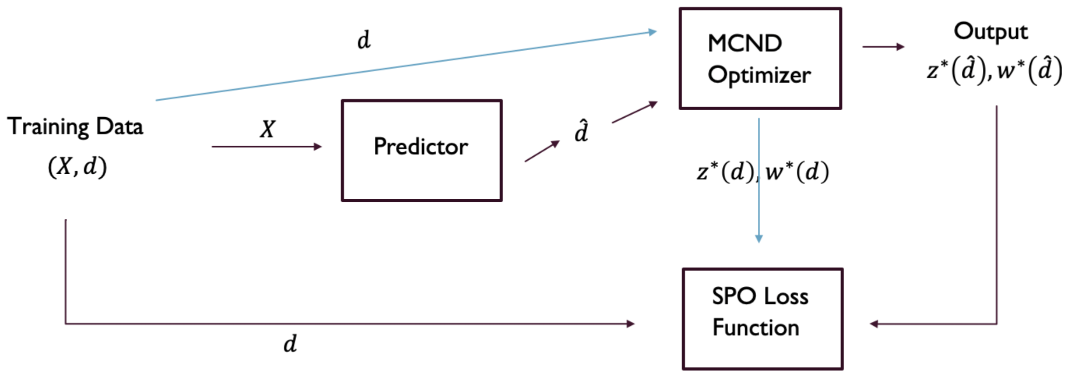


Figure 3.3 – Methodology Pipeline - This figure illustrates the pipeline of the DFL training. Training data (where d has already been run through the optimizer to obtain $z^*(d)$ and $w^*(d)$) gets fed into the predictor.

CHAPTER 4

RESULTS

This chapter presents an analysis and evaluation of the application of DFL in the context of MCND. It is an exploratory study where we aim to test the SPO framework on a problem it was not originally designed for. In Chapter 3, we outline the methodology. This chapter aims to showcase the outcomes and findings obtained from implementing the SPO framework, providing insights into its effectiveness and impact on network design outcomes.

The primary objective of this study is to explore how SPO performs on a MILP where unknown parameters are not in the objective function. We want to see if DFL can contribute to the improvement of network design, ultimately leading to optimized network designs, enhanced performance, and cost-effectiveness. To achieve this, a series of experiments, using the SPO loss functions to address simple instances of the MCND.

This chapter will go over the experimental setup, training, and testing of algorithms. We also compare SPO results with PO in different settings, high capacity, low capacity, and varying variable cost. Furthermore, this chapter discusses the significance of the results obtained, providing insights into the level of improvement achieved by using SPO loss functions in network design. Additionally, any limitations or challenges encountered during the implementation of SPO will be discussed as well.

4.1 Experimental Setup

The experimental setup will be important in order to outline the differences of SPO. We aim to compare how the predicted demands, predicted using different algorithms, affect the overall costs of networks. For this, we aim to use a small network that will allow a closer look into how arc selection, commodity flow, and other factors change with different algorithms. The metric that we use is also important in the analysis of results.

The overall objective is to study if SPO is able to select optimal arcs that lead to reduced costs compared to a baseline. This would show that SPO learns some signal of how to best solve MCND instances. We introduce the metric in Section 4.1.2.

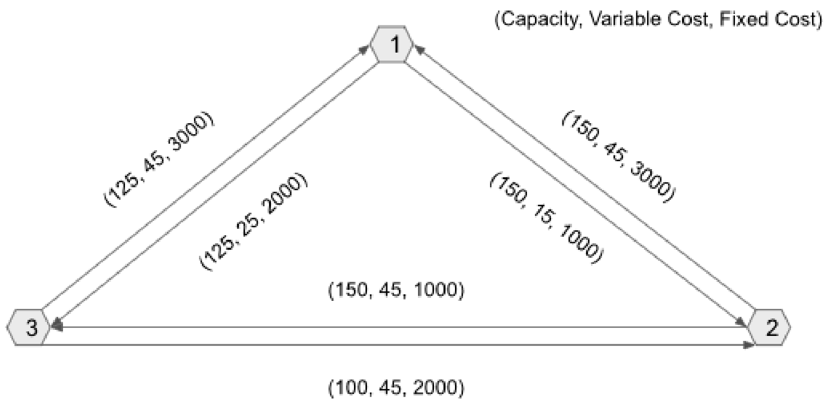


Figure 4.1 – Network Illustration - Illustration of the network with arc capacities, fixed costs, and variable costs.

The results are based on a very small network whose specifications are in Table 4.I. We also include an illustration of the network in Figure 4.1. The reasons we use a small network are two-fold. Firstly, the computation time required for DFL is significant and increases exponentially as can be seen in 3.I. A faster run-time helps broader experimentation of SPO as we were able to test more configurations of the model. Table 4.I shows the exact parameters of the network to be used.

Experiment	# Nodes	# Arcs	# Commodities	Demand Matrix Size
Test Design	3	6	3	3x3

Table 4.I – Test/Train Instance Specification - Outlines the size of the network used for results.

For more specific details about the exact network including arc capacities, variable costs, and fixed costs, see Appendix Table I.I.

The setting of the experimentation also requires a prediction model to assess the different loss function approaches. As this is an exploratory setting, we want to use the simplest predictor possible such that the explainability of the model is straightforward. This is also a regression task so the model must be capable. We selected linear regression with SGD for parameter estimation as it reduces training time and allows for straightforward analysis of inputs, outputs, and parameters, unlike many other ML algorithms that act as a black box.

This experimental setting allows for efficient and robust computations. In addition, the small network size allows for in-depth analysis of the following results.

4.1.1 Training

The training of SPO+ and PO follow the same steps. With the generated dataset, we train algorithms to minimize their respective loss functions outlined in Section 3 with a gradient descent algorithm. PO attempts to minimize prediction error while SPO+ minimizes a combination of prediction error and objective value outlined in Equation (2.9). Prior to training SPO+, we also compute the optimal solution value with true demand, denoted $z^*(c)$, as required in the SPO+ loss (2.9). We calculate optimal solutions for each data point, as required by the SPO+ framework. The model parameters are randomly initialized and then estimated using stochastic gradient descent to find the parameters that minimize the loss as explained as in Algorithm 1. We run five epochs of training data, as outlined in [30], which completes the training of the model.

4.1.2 Testing

We aim to test SPO on its ability to lower the overall objective value of the MCND while keeping accurate demand predictions. As such, we determine if the SPO loss function enables the signal from the MCND to make designs that reduce the overall objective value.

We split the data into training and testing at an 8:1 ratio. The training data is only used for training the SPO and PO algorithms. The testing set is unseen in training and is what is presented in this chapter.

In the results, we show three different models. The first is the baseline. We evaluate the models as follows. First, we compute the objective value using true demand, $z^*(d)$, which is the baseline. The two other models are PO and SPO+. Given the goal is to compare the different designs of SPO+ and PO given a certain data point, we compare the objective values and regret for each design. We denote the design using a predicted demand as $y^*(\hat{d})$. The regret we use in these results is shown in Equation (4.1) and differs from the regret used in the SPO+ loss function in Equation (3.1), which does not fix designs.

$$r_{mod}(d_i, \hat{d}_i) = z^{y^*(d_i)}(d_i) - z^{y^*(\hat{d}_i)}(d_i) \tag{4.1}$$

Essentially, given a predicted demand, $\hat{d}_i$, from either model, we calculate $y^*(\hat{d}_i)$. We then fix $y^*(\hat{d}_i)$ for the MCND (1.1) and use the true demand, d_i to evaluate. This test gives a realistic performance metric for both algorithms.

We test the different models in three different network capacity settings. The network capacity settings are meant to show cases where SPO does well and poorly and to help analyze results. The amount of capacity in the network has a big impact on performance. If capacity is small, predicting higher demands will result in much higher costs as the use of auxiliary arcs is necessary. There is an expectation that SPO will make predictions that do not use auxiliary arcs as it will severely penalize its loss function. If the capacities are large, the auxiliary arcs should not play a large role. On the other hand, with high variable costs, SPO will limit the amount of commodity it tries to predict to lower costs, with a trade-off of high prediction error. With no variable cost, there is no constraint in over-predicting, to the limit of using the auxiliary arcs. We present the results in the next three subsections.

4.2 Low Capacity with Variable Costs

Testing low capacity in a network design problem within the context of SPO serves several important purposes. Evaluating network designs with low capacity allows for an exploration of the network designs of SPO algorithms under constrained conditions. In addition, evaluating low-capacity arcs also allow for a comprehensive examination of trade-offs between different objectives in network design. With limited capacity, network designs are often forced to make strategic decisions on which arcs to use to avoid using very expensive overflow options. The capacities used in the section are outlined in Table I.I in the appendix.

Table 4.II shows the overall performance of the baseline, PO, and SPO. The mean objective values correspond to $z^{y^*(d_i)}(d_i)$ for the respective $\hat{d}$ averaged over the test set. The mean regret corresponds to $r(\hat{d}, d)$ (4.1) averaged over the test set. The mean number of arcs opened is the sum of arcs in design $y^*(\hat{d})$ average over the test set. The mean fixed cost is the sum of the fixed costs associated to the design, $y^*(\hat{d})$, averaged over the test set. Finally, the mean overflow units is the number of units that use the auxiliary costs when solving the MCND (1.1) with true demand, d and design, $y^*(\hat{d})$.

Table 4.III also shows the differences in the actual demand prediction each model is making. To clarify, we calculate the prediction error and regret of the actual predictions the PO and SPO models are making, where the regret follows Equation (3.1). In the table, we see that SPO has a negative regret meaning it is actually lower than the baseline objective cost. But, we see very large prediction errors meaning that SPO is under-predicting demand as a means to lower cost.

It is clear that the baseline has the lowest objective value (lower bound) with SPO being the highest. The same is true for regret. Moreover, the prediction error in Table 4.III

Design Metrics	Baseline	PO	SPO
Mean Objective Value	301,789	306,077	444,566
Mean Regret r_{mod}	0	4,288	142,777
Mean Number of Arcs Open	2.8	2.8	3.0
Mean Fixed Cost	5891	5891	5,000
Mean Overflow Units	288.7	288.7	433.3

Table 4.II – Network Design Metrics for Scenario 1 - Design metric results of Low Capacity, Variable Costs simulation.

ML Metrics	PO	SPO
Mean MSE for Demand Prediction	345	18,446
Mean Regret	586	-296,137

Table 4.III – ML Prediction Metrics for Scenario 1 - ML metric results of Low Capacity, Variable Costs simulation.

follows the same trend, where SPO has a mean prediction error significantly larger than PO.

This explains why SPO has a much higher regret than PO in Table 4.II. The design is meant for demands that are much lower than true demand which leads to the use of the very expensive auxiliary costs.

If we look at the way the gradients are calculated in Algorithm 1, we can see why the predictions tend to under-predict demand. The sub-gradients are computed using $w^*(d) - w^*(2\hat{d} - d)$ as in (3.8). The sub-gradients are being used to lead us parameters, θ, that minimize the ERM, effectively minimizing the SPO loss function (2.9). Essentially, the sub-gradients are computed by subtracting the decision variables given by solving the optimization problem with a demand equal to $2\hat{d} - d$ from the decision variables given by solving the optimization problem with the true demand, d. If $\hat{d}$ is larger than d, then $w^*(d) \leq w^*(2\hat{d} - d)$ and ∇L increases. On the other hand, if $\hat{d}$ is smaller than d, then $w^*(d) \geq w^*(2\hat{d} - d)$ and ∇L decreases. Since the objective of SGD is for the gradients to find the local minimum, SPO incentives $\hat{d}$ to decrease, which is what we see in the example. This relates to an issue with the objective function that we discuss further in Section 4.5.

In addition, when we compare model designs, PO and baseline have the same fixed cost and design. The small difference in objective value comes from the prediction error PO makes in demand. PO slightly over-predicts demand such that the variable costs of moving more commodities are higher. SPO selects more arcs in the design yet has a lower fixed cost.

Figure 4.5 shows the average demand prediction of a commodity per node. We that the PO predictions are usually closer to actual demand than SPO. This also shows that

SPO is under-predicting demands.

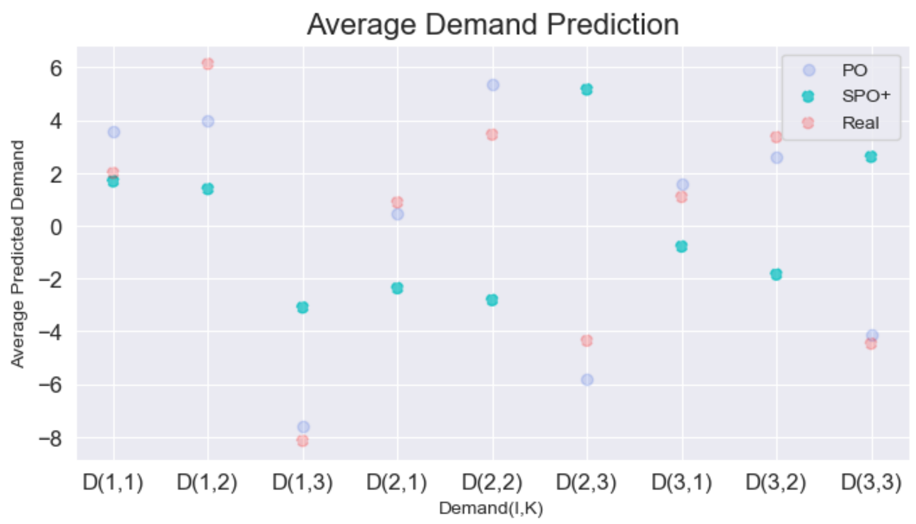
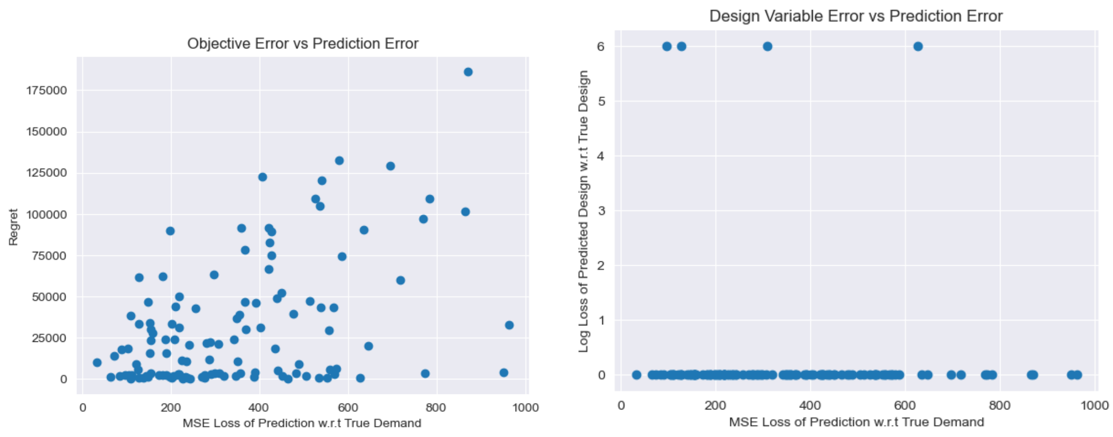


Figure 4.2 – Demand Prediction for Scenario 1 - Average demand prediction for each commodity.

4.2.1 Predict, then Optimize

Figure 4.3a and 4.3b show the prediction error versus regret and prediction error versus design variable error. It shows that the prediction error and regret are uncorrelated. Generally, the trend we see is; the higher the prediction error, the lower the regret. Since the predictive algorithm does not learn from the MCND structure, we expect good prediction results, especially considering the linearly constructed data.

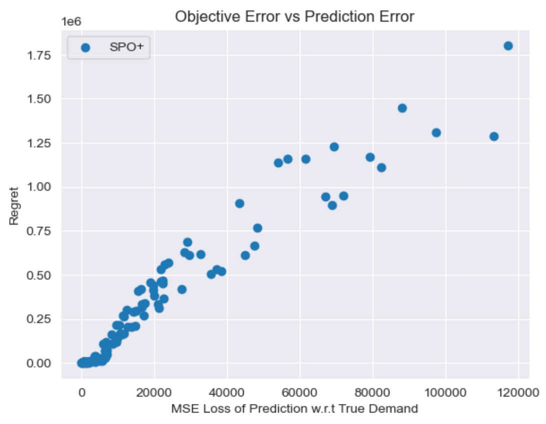


(a) Regret and Prediction Error (b) Variation in Design

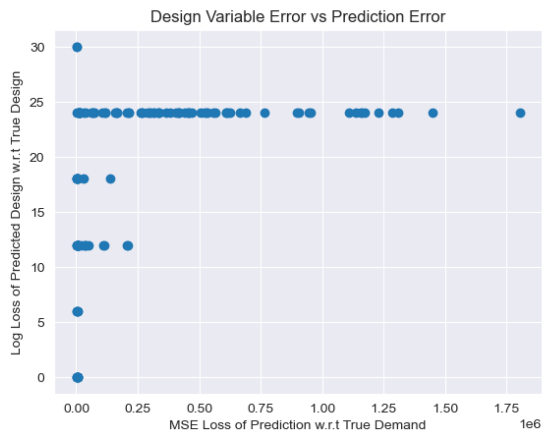
Figure 4.3 – Objective and Design Graphs for Scenario 1, PO - The left graph shows the relationship between Regret and MSE Prediction Error. The right graph shows the relationship between the difference in design variable selection and MSE prediction error.

We also see that there is little variation in the design variables. Almost all generated demands are being solved with the same design. However, this can almost certainly be attributed to the low capacity with respect to the amount of flow. Since every arc is being utilized near full capacity, there is not much room for difference in arc selection leading to the results shown. This example shows the importance of capacity in the MCND problem.

4.2.2 Smart Predict then Optimize



(a) Regret and Prediction Error



(b) Variation in Design

Figure 4.4 – Objective and Design Graphs for Scenario 1, SPO - The left graph shows the relationship between Regret and MSE Prediction Error. The right graph shows the relationship between the difference in design variable selection and MSE prediction error.

SPO differs from the PO. We see that for the framework, there is a correlation between regret and predictive error. The two metrics are quite correlated where higher prediction error leads to higher regret. Yet, one thing to notice is the degree of the predictive error as outlined in Table 4.III. Note that we experimented with changing true demand values and this does not change the overall performance of this approach.

It is also apparent in Figure 4.4b that SPO uses various designs, as is apparent from the different log losses of the designs. We also see that the designs include only 3 arcs and have the same fixed cost in Table 4.II. This shows that the SPO loss function is able to actually gain signal from the MCND. While it is unable to accurately predict demand, it does lower fixed costs (5,000 vs. 5,891 for SPO and PO, respectively). Yet, the lower fixed costs end up being extremely costly when using true demand as it requires a high use of auxiliary arcs. As SPO is under-predicting demand, to lower initial fixed costs that are ideal with its small demand predictions, it is not able to consider that its savings lead to much higher auxiliary costs when using true demand.

The SPO loss function penalizes large demand predictions. What we see in this example is that the computation of sub-gradients leads to under-predictions and favors reducing the costs rather than prediction error. In this capacity setting, the SPO loss function is not appropriate as shown by bad prediction and large regret.

4.3 High Capacity with Variable Costs

Testing high capacity in the MCND is equally important as it provides valuable insights into the design variable selection. By evaluating network designs with high capacity, we can examine how the SPO algorithm effectively utilizes abundant capacity to optimize network performance. In addition, testing high-capacity arcs enables us to analyze the decision-making capabilities of DFL. Since there is more capacity, we will observe different arc selections between algorithms. The capacity used the number outlined in Table I.I, in the appendix, multiplied by a factor of 10.

We see a similar trend here as to low capacity. It is clear that in Table 4.IV, SPO has a large prediction error but improves the regret even compared to baseline (the demand predictions are small which causes lower objective values when comparing to baseline). This is due to underpredictions caused by the definition of loss function.

Table 4.IV shows the baseline and PO models actually have a lower fixed cost but a higher auxiliary cost compared to SPO. We also notice that all the models return lower objective values compared to the smaller capacity example due to avoiding the large auxiliary arc costs. This could mean that the SPO does gain some signal into how to lower network costs. However, the overall objective cost of SPO is still higher. We explore this further in the next sub-sections.

Design Metrics	Baseline	PO	SPO
Mean Objective Value	21,827	24,791	29,780
Mean Percent of Arcs Open	36 %	36%	50%
Mean Fixed Cost	4,033	4,033	5,000
Mean Overflow Units	3,304	3,304	1,359

Table 4.IV – Network Design Metrics for Scenario 2 - Design metric results of Low Capacity, Variable Costs simulation.

ML Metrics	PO	SPO
Mean MSE for Demand Prediction	345	18,430
Mean Regret	-520	-16,331

Table 4.V – ML Prediction Metrics for Scenario 2 - ML metric results of Low Capacity, Variable Costs simulations.

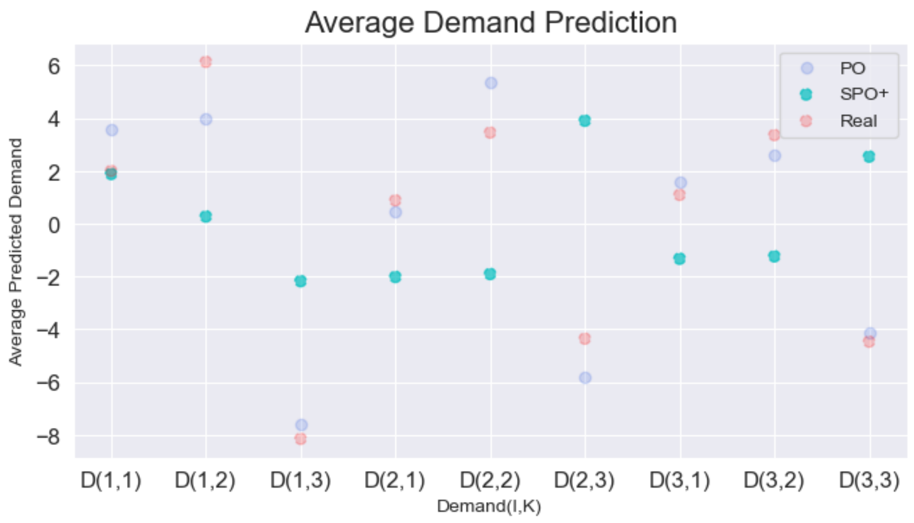
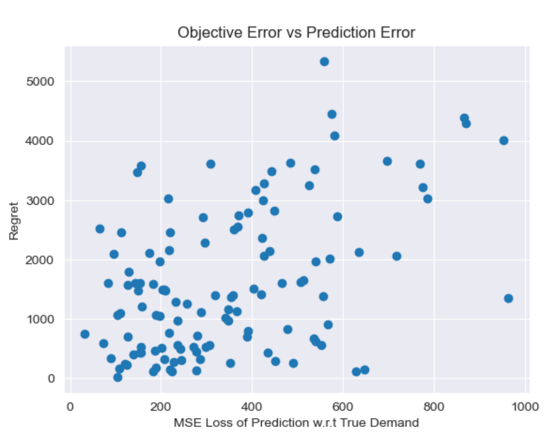


Figure 4.5 – Demand Prediction for Scenario 2 - Average demand prediction for each commodity.

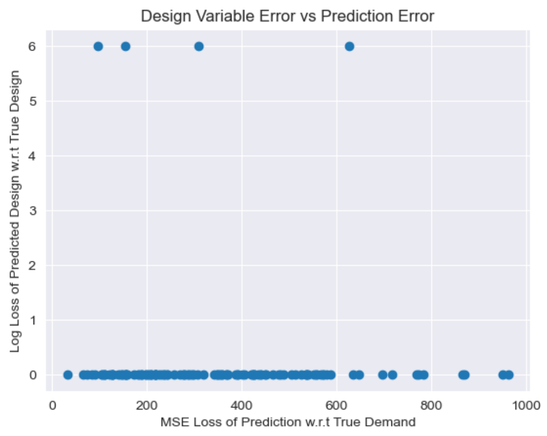
Figure 4.5 shows the same illustration as previously mentioned. We see here, SPO demand prediction usually underpredicts with respect to the baseline.

4.3.1 Predict, then Optimize

PO has the same results as Section 4.2. Similarly, it can also be attributed to small prediction errors. The takeaway is that when models predict demands accurately, the regret stays very small as well.



(a) Regret and Prediction Error

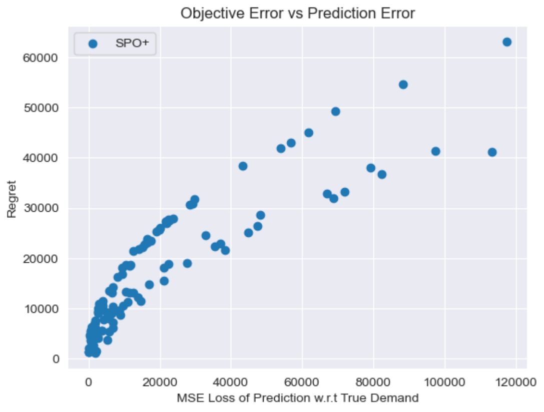


(b) Variation in Design

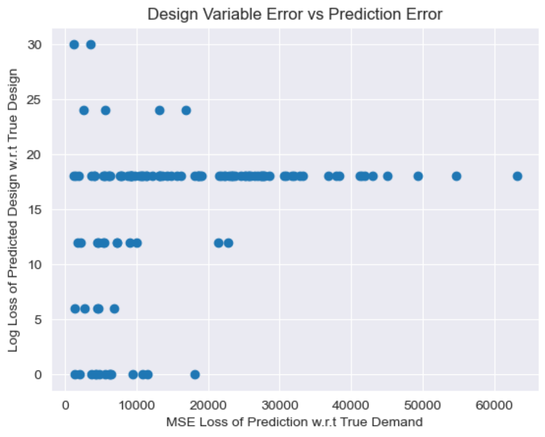
Figure 4.6 – Objective and Design Graphs for Scenario 2, PO - The left graph shows the relationship between Regret and MSE Prediction Error. The right graph shows the relationship between the difference in design variable selection and MSE prediction error.

4.3.2 Smart Predict then Optimize

In contrast, it is easy to see that SPO yields different results. In Figure 4.7a, prediction error and regret are correlated. We see that the demand predictions SPO makes are still underpredicting, yet we see a small decrease in predictive error. Once again, we see that SPO underpredicts to get small objective values but makes large prediction errors.



(a) Regret and Prediction Error



(b) Variation in Design

Figure 4.7 – Objective and Design Graphs for Scenario 2, SPO - The left graph shows the relationship between Regret and MSE Prediction Error. The right graph shows the relationship between the difference in design variable selection and MSE prediction error.

4.4 High Capacity, No variable costs

Testing a network design problem with no variable cost is to explore the impact of fixed cost. We hypothesize that in this scenario, demands will be accurately estimated as there is no penalty for overestimating as the only cost consideration would be the fixed costs associated with opening arcs. This allows us to gain a deeper understanding of how SPO prioritizes and optimizes network designs when the variable cost is not a driving factor.

Design Metrics	Baseline	PO	SPO
Mean Objective Value	6,105	6,105	6,359
Mean Percent of Arcs Open	35%	35%	50 %
Mean Fixed Cost	3,208	3,208	5,000
Mean Sum of Auxiliary Cost	2,897	2,897	1,359

Table 4.VI – Network Design Metrics for Scenario 3 - Design metric results of Low Capacity, Variable Costs simulation.

We see a different trend in Table 4.VI for this example. All objective functions are similar. SPO has a lower auxiliary arc cost due to more arcs being available. The tradeoff is that the fixed costs of opening the arcs are larger. PO and baseline have higher auxiliary costs and lower fixed costs due to fewer arcs, in contrast.

ML Metrics	PO	SPO
Mean MSE for Demand Prediction	358	18,300
Mean Regret	135	1,637

Table 4.VII – ML Prediction Metrics for Scenario 3 - ML metric results of Low Capacity, Variable Costs simulation.

We see different results in Table 4.VII as well. Regret and prediction error is small for PO while SPO makes large prediction errors but reduces objective value with its underpredictions.

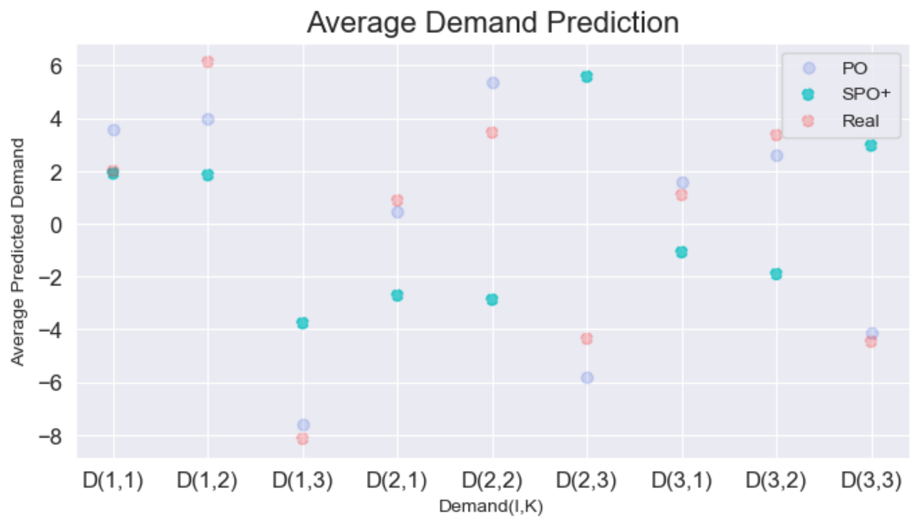
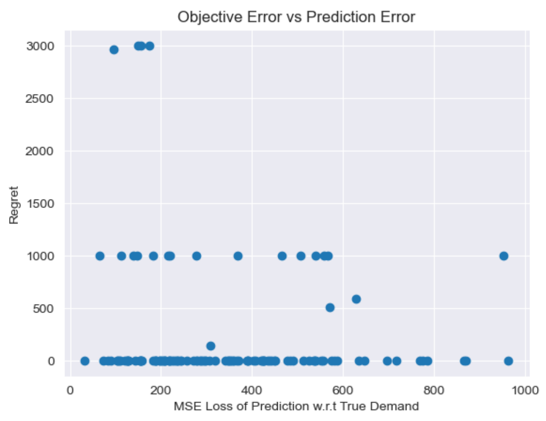


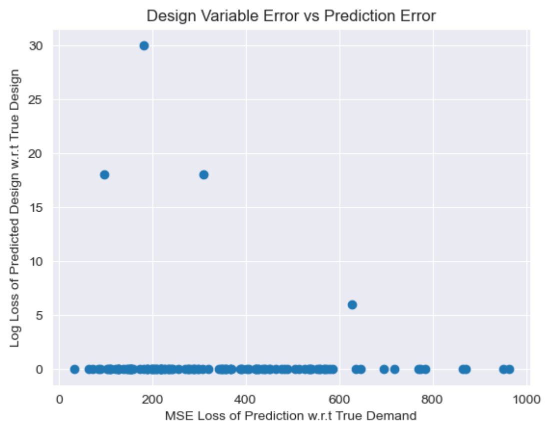
Figure 4.8 – Demand Prediction for Scenario 3 - Average demand prediction for each commodity.

4.4.1 Predict, then Optimize

Due to no variable cost, we can see in Figure 4.9a that the resulting prediction error and regret are on discrete points representing the various initial fixed costs.



(a) Regret and Prediction Error



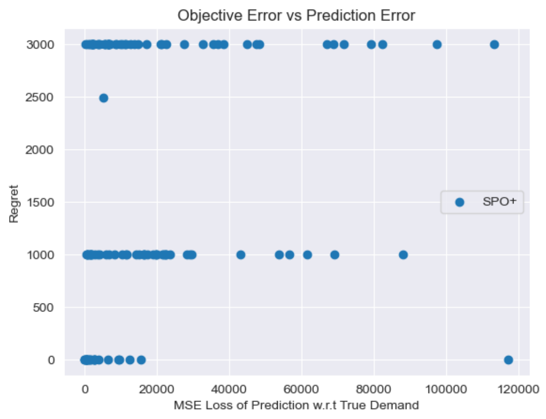
(b) Variation in Design

Figure 4.9 – Objective and Design Graphs for Scenario 3, PO - The left graph shows the relationship between Regret and MSE Prediction Error. The right graph shows the relationship between the Difference in Design Variable Selection and MSE Prediction Error.

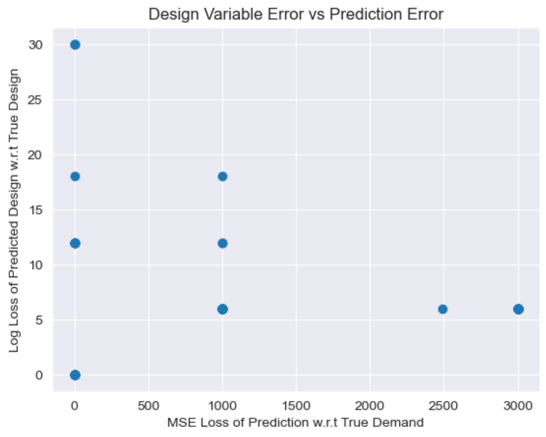
PO performs baseline in this example. Sie prediction error is small and they have the same fixed cost, we expect total costs to be the same as there is no variable costs.

Figure 4.9b has the same behaviour as in Sections 4.2 and 4.3 where designs remain generally constant.

4.4.2 Smart Predict then Optimize



(a) Regret and Prediction Error



(b) Variation in Design

Figure 4.10 – Objective and Design Graphs for Scenario 3, SPO - The left graph shows the relationship between Regret and MSE Prediction Error. The right graph shows the relationship between the difference in design variable selection and MSE prediction error.

We see similar results as PO in Figure 4.10a. Since SPO has more arcs opened, we see a higher fixed cost but lower auxiliary costs, which is expected. There is no correlation between prediction error and regret here. In a situation with high capacity and no variable cost, SPO and PO perform similarly.

Figure 4.10b shows that designs fluctuate but are not correlated to predictions errors.

4.5 Discussion

The takeaway of the results is two-fold. First and foremost, SPO performs below the standard PO in this instance. It is clear to see that SPO is outperformed by PO in reducing regret (4.1) and prediction errors. The designs that PO makes have lower objective values and select fewer arcs in the design on average. As SPO model underestimates demand, it selects designs that, when using true demand, rely on auxiliary arcs, especially in low-capacity settings. It is apparent that the SPO loss function underestimated demand to prediction regret (3.1), showing it cares more about reducing the objective value than accurate predictions. In real examples, clients' need for accurate demand prediction is important and models that under-predict lead to lower customer satisfaction and over-prediction leads to unnecessarily high costs. Through all examples, the SPO loss function prioritizes lowering prediction regret over accurate predictions which is not a trade-off SPO should make in this example. Moreover, the resulting designs showed that using the network design from SPO resulted in higher regret.

The second important point to make about these results is that SPO is not sensitive enough to the different capacity parameters. This signals a flaw in the use of SPO for the MCND. From Tables 4.III, 4.V, and 4.VII it is apparent that regret changes according to capacity settings but the prediction error of SPO only fluctuates by a small amount. The prediction errors of PO remain constant throughout all settings which is expected as it only accounts for prediction error and is unchanged by the downstream optimization problem.

The SPO prediction errors are 18,446, 18,430, and 18,300 for low capacity, high capacity, and high capacity with no variable cost, respectively. This is the trend expected as with low capacity, the SPO loss function (2.9), is incentivized to under-predict as over-prediction will result in the use of expensive auxiliary arcs. With high capacity, predictions are reduced slightly as there is less of a dependency on auxiliary arcs. With high capacity and no variable cost, we expect the prediction to drop more as there are no additional costs associated with having more commodity flow. Even though we see that the SPO model does gain some signal from the optimization problem, it still under-predicts the demand and requires further customization.

There is one important note to make. For this experiment, we used linearly generated data with an i.i.d assumption. This is far from reality as time-dependent data is much harder to predict and features are far from linear. This leads to a high performance of the PO algorithm which does not perform as well in more realistic settings.

MCND is a fundamentally different problem than the ones SPO was designed for. We relax MCND such that the unknown parameters are in the objective function, which allows the use of the SPO framework. This means that the regret will drive demand down and impact both the *feasible region* and objective costs. From the results, SPO in this original version (designed for cost parameters only) is not appropriate for this case.

As noted, SPO is not designed for the MCND. Throughout this thesis, adjustments are made to nevertheless explore the use of SPO in this setting. We believe that more customization could lead to better results of DFL applied on the MCND.

CHAPTER 5

CONCLUSION

This thesis focused on the exploration of DFL in network design. We introduced the MCND with easy-to-understand description and notation. Furthermore, we delve into the current literature of network design, ML and DFL, highlighting what has been done but also bringing to attention areas that need development.

One of the issues we encountered in this work is the large computational resources required to perform SPO on the MCND. This constrained the work to smaller networks and linear data generation to find convergence in training and testing algorithms.

The work serves to achieve two goals. We first provide a motivating example that serves to bring to attention the potential benefits of SPO in the context of the MCND problem. We wanted to investigate, with the modifications shown in this thesis, if SPO could solve a problem it was not designed for. The amount of research into DFL is increasing but it lacks the applicability to many problems.

Secondly, we provided the exploration of using SPO to design networks in the MCND. We outlined the computational difficulties in our approaches. The main takeaway from the experimental result was that with the customization we do, PO performs better. However, we believe that with further methodological advancements of a loss function, results could improve from the ones we present here. It is clear that this implementation is artificially underpredicting demand such that it designs networks that increase operation costs due to its reliance on overflow penalties for realized demands. The underprediction of demand stems from the SPO loss function that focuses on decreasing regret more than prediction errors in this setting, SPO does learn how to reduce prediction regret, the error in prediction values is too significant and leads to network designs that are outperformed by PO, due to the feasible region being impacted. We think that future research should look at loss function modifications that can further minimize prediction loss while keeping the low downstream costs we showed here.

In conclusion, this thesis accomplished the goal of exploring SPO in the context of the MCND. Additionally, the work presents preliminary exploratory results and highlights future works in the area.

BIBLIOGRAPHY

[1] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. (2019). Differentiable convex optimization layers. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[2] Amos, B. and Kolter, J. Z. (2017). OptNet: Differentiable optimization as a layer in neural networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR.

[3] Bengio, Y. (1997). Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, 08(04):433–443.

[4] Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. (2020). Learning with differentiable pertubed optimizers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9508–9519. Curran Associates, Inc.

[5] Bertsimas, D., Brown, D. B., and Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501.

[6] Canada (2022). Environmental indicators - gouverment of canada. https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/greenhouse-gas-emissions.html.

[7] Chouman, M., Crainic, T. G., and Gendron, B. (2017). Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, 51(2):650–667.

[8] Cox, D. R. and Hinkley, D. V. (1974). *Theoretical Statistics*. Chapman & Hall, London, England.

[9] Crainic, T.G., G.-B. . H. G. (2004). A slope scaling/lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics*, 10:525–545.

[10] Crainic, T. G., Frangioni, A., and Gendron, B. (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):73–99.

[11] Crainic, T. G., Gendreau, M., and Farvolden, J. (2000). A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, 12:223–236.

[12] Donti, P., Amos, B., and Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 5484–5494.

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

[14] Fisher, M. L. (1981). The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18.

[15] Frejinger, E. and Hewitt, M. (2021). Perspective on optimizing transportation systems with interacting supplies and demands. *Submitted to Transport Science*.

[16] Gendron, B., Crainic, T. G., and Frangioni, A. (1999). Multicommodity capacitated network design. *Telecommunications Network Planning*, page 1–19.

[17] Gendron, B., Hanafi, S., and Todosijević, R. (2018). Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research*, 268(1):70–81.

[18] Gendron, B. and Larose, M. (2014). Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization*, 2(1):55–75.

[19] Glasmachers, T. (2017). Limits of end-to-end learning. *Proceedings of Machine Learning Research*.

[20] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[21] Gurnani, H. and Tang, C. S. (1999). Note: Optimal ordering decisions with uncertain cost and demand forecast updating. *Management Science*, 45(10):1456–1462.

[22] Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

[23] Hewitt, M., R.-W. and Wallace, S. (2022). *Service Network Design. Chapter 10: Stochastic Network Design*.

[24] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[25] Kotary, J., Dinh, M. H., and Fioretto, F. (2023). Backpropagation of unrolled solvers with folded optimization. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

[26] Kotary, J., Fioretto, F., Van Hentenryck, P., and Wilder, B. (2021). End-to-end constrained optimization learning: A survey. *Proceedings of the Conference on Artificial Intelligence*.

[27] Laage, G., Frejinger, E., and Savard, G. (2021a). Periodic freight demand forecasting for large-scale tactical planning. *CoRR*, abs/2105.09136.

[28] Laage, G., Frejinger, E., and Savard, G. (2021b). A two-step heuristic for the periodic demand estimation problem.

[29] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[30] Mandi, J., Demirovic, E., Stuckey, P. J., and Guns, T. (2019). Smart predict-and-optimize for hard combinatorial optimization problems. *CoRR*, abs/1911.10092.

[31] Mills, T. C. (2015). *Time series techniques for economists*. Cambridge University Press.

[32] Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Bucarey, V., and Guns, T. (2020). Contrastive losses and solution caching for predict-and-optimize. In *International Joint Conference on Artificial Intelligence*.

[33] Paulus, A., Rolínek, M., Musil, V., Amos, B., and Martius, G. (2021). Comboptnet: Fit the right np-hard problem by learning integer programming constraints. *Proceedings of Machine Learning Research*.

[34] Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., and Vidal, T. (2023). A survey of contextual optimization methods for decision making under uncertainty. *CoRR*.

[35] Stock, J. and Watson, M. W. (2003). *Introduction to Econometrics*. Prentice Hall, New York.

[36] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, abs/1706.03762.

[37] Winston, C. (1983). The demand for freight transportation: models and applications. *Transportation Research Part A: General*, 17(6):419–427.

[38] Wu, N., Green, B., Ben, X., and O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *CoRR*, abs/2001.08317.

[39] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115.

Appendix I

Information on Network Settings

Arc	Fixed Cost	Variable Cost	Capacity
(1, 2)	1,000	15	150
(1, 3)	2,000	25	125
(2, 1)	3,000	45	150
(2, 3)	1,000	45	150
(3, 1)	3,000	45	125
(3, 2)	2,000	45	100
Auxiliary Arcs			
(1, 2)	-	1,000	-
(1, 3)	-	1,000	-
(2, 1)	-	1,000	-
(2, 3)	-	1,000	-
(3, 1)	-	1,000	-
(3, 2)	-	1,000	-

Table I.I – Detailed Information on Test Network Instance - Outlines the specifications of the network used for all the results.