APPROXIMATE ROBUST OPTIMIZATION ALGORITHM AND NUMERICAL INVESTIGATION

Quan Zhou*
Desautels Faculty of Management
McGill University
Bronfman Building, Room 521
Montréal, QC H3A 1G5, CA
quan.zhou3@mail.mcgill.ca

Luiz Resende Silva†
Department of Mathematics and Industrial Engineering
École Polytechnique de Montréal
Pavillon Principal, Bureau A-520.28
Montréal, QC H3T 1J4, CA
luiz.resende-silva@polymtl.ca

ABSTRACT

A robust optimization framework is a practical tool for decision-making when facing uncertainty given its tractability and no need for a priori probability distribution. However, the increasing computational complexity and the requirement of efficient solvers limit its usage in large-scale problems. Several iterative methods are proposed to address that issue. In this paper, we carried out a numerical investigation to an oracle-based approximate robust optimization algorithm based on Ben-Tal et al. (2015b) in an advertisement campaign problem. We show that the algorithm has sublinear convergence rate and is able to return a 2ϵ-approximate optimal solution after a finite number of iterations. When the problem size increases such that the nonlinear optimization solver failed, the algorithm was still capable to solve such problem instance without deteriorating running time.

1 INTRODUCTION

The Robust Optimization (RO), which seeks a solution that is robust to all possible realizations of uncertainty parameters from a known set, is one of the leading paradigms for optimization problems under uncertainty. We refer the readers to the seminal papers by Ben-Tal & Nemirovski (2002); Ben-Tal et al. (2009); Bertsimas et al. (2011) for detailed RO theory and numerous applications.

Despite its theoretical and empirical success, it is still challenging to apply RO to large-scale problems due to increased computational complexity, which limits the usage of RO in machine learning context for more robust and stable solutions. The classical approach for solving a RO problem is done through reformulating the problem using duality theory and then solving the reformulated robust counterpart (RC) with a commercial solver. However, such RC reformulation is often not as scalable and easy to solve as the deterministic nominal problem, and sometimes a special solver may be required.

Alternatively, an iterative algorithm that generates a candidate solution and a new uncertainty realization in turn offers a remedy to the scalability issues associated with the classic RO approach. Researchers have worked in this direction and proposed several oracle-based iterative algorithms that generates approximate solutions to the original RO problem (Calafiore & Campi, 2005; Mutapcic & Boyd, 2009; Ben-Tal et al., 2015b).

*https://github.com/qzhou-2020
†https://github.com/luiz-resende

It is of interest form a researching point to investigate how such iterative algorithms can be implemented and used in solving real problems. Therefore, in this paper we followed one of the seminal papers by Ben-Tal et al. (2015b) and implemented the oracle-and-gradient descent based algorithm with modifications. We then apply this algorithm to an advertisement campaign problem and compared the performance with the classical approach based on Fenchel robust counterpart. Our numerical study revealed that, with mild modifications, the approximate algorithm was able to reach the accuracy requirement within a finite number of iterations at *sublinear* convergence rate. The approximate algorithm is reliable, even in those cases when a nonlinear optimization solver failed to handle the reformulation in the classical approach. However, due to the computational complexity of oracle, we observe a linear time increase in the approximate method as the dimension of the problem increases.

2 Related works

The approximate method implemented in this paper is closely related to online learning (Cesa-Bianchi & Lugosi, 2006), online convex optimization (Shalev-Shwartz et al., 2012; Hazan et al., 2016), and adversarial approach for robust optimization (Gorissen et al., 2015). We refer the readers to those papers and books for more detailed discussion about those topics. We limit ourselves to the topic of robust convex optimization problems and briefly introduce the developments in this direction.

Calafiore & Campi (2005) proposed a "constraint sampling" approach by independently and identically distributed sampling of uncertainty parameter realizations and forming a "extended" nominal problem of the same class as the original one. They demonstrated that the solution to this extended nominal problem is robustly feasible with high probability to the original robust optimization, where the probability depends on the sampling procedure, the number of samples, and the dimension of the problem.

Mutapcic & Boyd (2009) follow a "cutting-plane"-type approach that relies on solving two oracles in each iteration. A similar iteration strategy is adopted in the column-and-constraint generation method for the two-stage adjustable robust linear programming (Zeng & Zhao, 2013). The first oracle solves a similar extended nominal problem for Calafiore & Campi (2005) to obtain a feasible solution. The second oracle, referred to as *pessimization oracle*, is invoked to expand and refine the extended nominal problem. Provided with a candidate solution, the pessimization oracle either certifies its feasibility with respect to all of the robust constraints (robust optimality) or return a new realization of the uncertainty parameters from the uncertainty set for which the candidate is infeasible. If such a realization is returned, it will be added into the extended nominal problem. The process is repeated until a robust optimal solution is reached or the last extended problem is infeasible. The computational complexity heavily depends on solving the two oracles, as well as the number of iterations. The author reported that the number of iterations can be exponential in the dimension.

Both methods mentioned earlier have troubles in solving high-dimensional problems because of the number of constraints in the extended nominal problem. In Calafiore & Campi (2005), one has to sample more uncertainty realization to ensure the high-probability guarantee as the dimension increases. This results in a linearly increase in the number of constraints. In Mutapcic & Boyd (2009), one new constraint is added to the extended nominal problem in each iteration, and theoretically the number of constraints can increase exponentially as the dimension increases.

To address the issue caused by solving the extended nominal problem with increasing number of constraints, Ben-Tal et al. (2015b) introduced a new iterative approach in which a candidate solution can be obtained by solving a nominal problem. The uncertainty parameter is updated with online gradient descent method (Zinkevich, 2003), and the new uncertainty realization is then used to update the nominal problem. Although the proposed approach still requires to solve an oracle, the nominal problem in the oracle has exactly the same complexity to the deterministic formulation of the original robust optimization problem and it is independent of the dimension and the number of iterations. Another distinguishing feature is that Ben-Tal et al. (2015b) replaced the pessimization oracle with an online gradient descent algorithm that only requires the first order information from the constraint functions.

Our work follows the oracle-based algorithm proposed by Ben-Tal et al. (2015b) with modifications. In particular, we solve an optimization problem (instead of a feasibility problem as in the original paper) using the oracle to guarantee the convergence of the objective function value. We also added an early stopping criteria and replaced the constant step-size with a convex decreasing step-size (Zinkevich, 2003).

The rest of this paper is organized as follows. In Section 3 we introduce the necessary techniques of online convex optimization, followed by the theoretical analysis and implementations of the oracle-based algorithm. In Section 4 we describe the Case Study framework used, the different uncertainty sets tested and the Fenchel Robust Counterpart (FRC) reformulations with each of those uncertainty sets. In Section 5 we compared the performances of directly solving the FRC to that of using the approximate algorithm. Finally, Section 6 summarizes the paper findings and discusses possible future extensions to this work.

3 APPROXIMATE ROBUST OPTIMIZATION

Consider the following robust optimization problem

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x, u_i) \leq 0, \quad \forall\, u_i \in \mathcal{U}_i,\, i = 1, \cdots, m, \\ & x \in \mathcal{D} \end{aligned} \tag{3.1}$$

where f_0 is a convex function in x, $f_1, \cdots, f_m$ are convex functions in x and concave functions in u_i, $\mathcal{D} \subseteq \mathbb{R}$ is a convex feasible set of x, the parameter vector $u = [u_1, \cdots, u_m]^T$ lies in an uncertainty set $\mathcal{U} = \mathcal{U}_1 \times \cdots \times \mathcal{U}_m$. The corresponding nominal problem with a known u is

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x, u_i) \leq 0, \quad i = 1, \cdots, m, \\ & x \in \mathcal{D} \end{aligned} \tag{3.2}$$

An approximate robust optimization solver deals with Equation (3.1) not through Fenchel robust counterpart but with an iterative approach (Mutapcic & Boyd, 2009; Ben-Tal et al., 2015b). At each iteration, Equation (3.2) is solved by an *oracle*, and an ϵ-approximate solution, which is in $\mathcal{D}$ and at most violates any constraint by ϵ, is obtained. Upon the termination of iteration, the solver produces an ϵ-approximate solution to the original problem in Equation (3.1).

The oracle-based approximate robust optimization algorithm is based on *regret* minimization in online convex optimization using *Online Gradient Descent* (OGD) algorithm. In the rest of the section, we will introduce both concepts followed by the implementation of the oracle-based approximate robust optimization algorithm.

3.1 REGRET AND ONLINE CONVEX OPTIMIZATION

Regret minimization is a technique used in *online learning framework* that deals with the following task: a predictor observes one after another element from a sequence, denoted as $x_1, x_2, \cdots$ at time $t = 1, 2, \cdots, x_t \in \mathcal{X}$. After the observation of x_t, it must guess a value for $\hat{y}_t \in \mathcal{Y}$ and then receive the actual y_t. The guessing goes on for T rounds and the goal for the predictor is to provide as "good" of a guesses as possible. Without the fundamental assumption that samples are generated by an underlying stochastic process, one cannot complete this task by minimizing the *risk*. Instead, researchers measure a predictor's performance by comparing it to a baseline model. Let's assume $L(\cdot, \cdot)$ denotes a *loss function* (Shalev-Shwartz et al., 2012), and define a regret as the cumulative difference in performance between the predictor and the baseline model, i.e.,

$$R(T) = \sum_{t=1}^{T} L(\hat{y}_t, y_t) - L(\tilde{y}_t, y_t)$$

In an online convex optimization problem, the convex set $\mathcal{S}$ is known in advance, but in each step of some repeated optimization problem, one must select a point $z^t \in \mathcal{S}$ before seeing the concave reward function[1] $r^t(\cdot)$ at that step. The goal is to "learn" the reward function such that the total reward over some steps T is maximized.

It is admissible to use a static policy as the reference model. Such a policy selects a single and fixed point from $\mathcal{S}$ for all steps. Mathematically, it is equivalent to solve the following optimization problem.

$$\max_{z \in \mathcal{S}} \sum_{t=1}^{T} r^t(z)$$

We then define the regret as

$$R_\pi(T) = \max_{z \in \mathcal{S}} \sum_{t=1}^{T} r^t(z) - \sum_{t=1}^{T} r^t(z^t) \tag{3.3}$$

where z^t is determined by a policy π. Our goal is to find this policy π such that the average regret $R_\pi(T)/T$ approaches to 0 as T increases.

Consider any ϵ-approximate constraint $f_i(x, z_i) \leq \epsilon$ for all $z_i \in \mathcal{Z}_i$ in Equation (3.1). It is equivalent to

$$\sup_{z_i \in \mathcal{Z}_i} f_i(x, z_i) \leq \epsilon.$$

The function $f_i(x, \cdot)$ is unknown before solving the original problem. However, if there exists a learner such that it's average regret is 0, this learner is able to guess an ϵ-feasible solution to Equation (3.1). Later, we will show that there is a way to generate an ϵ-optimal solution based on those ϵ-feasible solutions.

3.2 ONLINE GRADIENT DESCENT (OGD) ALGORITHM

We adopt the OGD algorithm proposed in Zinkevich (2003) as the policy for choosing $z \in \mathcal{Z}$. The OGD algorithm starts with an arbitrary $z^1 \in \mathcal{Z}$ and a sequence of step sizes $\{\eta_t\}$, $\eta_t \in \mathbb{R}^+$. After receiving the reward function at step t, it updates the iterate by

$$z^{t+1} = P(z^t + \eta_t \nabla r^t(z^t)) \tag{3.4}$$

where $P(\cdot)$ is an projection operator that ensures the next iterate is still in $\mathcal{U}$. Mathematically, the projection operator is equivalent to the following optimization problem

$$P(z) := \arg\min_{z' \in \mathcal{Z}} ||z' - z||_2 \tag{3.5}$$

$\nabla r^t(\cdot)$ denotes the gradient of the reward function r^t. We assume r^t is differentiable for all t and the gradient is upper bounded

$$G \geq \max_t ||\nabla r^t(z^t)||_2.$$

We also assume that the uncertainty set $\mathcal{U}$ is compact, and its diameter is finite

$$D \geq \max_{z, z' \in \mathcal{Z}} ||z' - z||_2$$

[1] We are interested in online maximization of concave function rather than minimization of convex function. Both formulations are equivalent.

Theorem 1 (Regret Bound, Zinkevich (2003)). *If $\eta_t = 1/\sqrt{t}$, the regret of OGD algorithm is*

$$R_{OGD}(T) \leq \frac{D^2\sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right) G^2$$

Therefore, $\limsup_{T\to\infty} R_{OGD}(T)/T = 0$.

Proof. Suppose $z^* := \arg\max_x \sum_{t=1}^T r^t(z)$. By concavity of the reward function r^t we have

$$r^t(x^t) - r^t(x^*) \geq \nabla r^t(x^t)(x^t - x^*)$$

which implies

$$\begin{aligned} R_{OGD}(T) &= \max_{z\in\mathcal{S}} \sum_{t=1}^T r^t(z) - \sum_{t=1}^T r^t(z^t) \\ &= \sum_{t=1}^T r^t(x^*) - r^t(x^t) \\ &\leq \sum_{t=1}^T \nabla r^t(x^t)(x^* - x^t) \end{aligned} \tag{3.6}$$

On the other hand, suppose

$$\tilde{z}^{t+1} = z^t + \eta_t \nabla r^t(x^t)$$

which implies

$$\begin{aligned} (\tilde{z}^{t+1} - z^*)^2 &= (z^t - z^*)^2 + \eta_t^2(\nabla r^t(z^t))^2 + 2\eta_t(z^t - z^*)\nabla r^t(z^t) \\ &\leq (z^t - z^*)^2 + \eta_t^2 G^2 + 2\eta_t(z^t - z^*)\nabla r^t(z^t) \end{aligned}$$

Notice that with the projection operation, the actual updated iterate in Equation (3.4) will not be further to z^* than $\tilde{z}^{t+1}$, i.e.,

$$(z^{t+1} - z^*)^2 \leq (\tilde{z}^{t+1} - z^*)^2$$

and hence,

$$(z^* - z^t)\nabla r^t(z^t) \leq \frac{1}{2\eta_t}\left[(z^t - z^*)^2 - (\tilde{z}^{t+1} - z^*)^2\right] + \frac{\eta_t}{2} G^2 \tag{3.7}$$

Combine Equation (3.6) and Equation (3.7) we have

$$\begin{aligned} R_{OGD}(T) &\leq \sum_{t=1}^{T} \nabla r^t(x^t)(x^* - x^t) \\ &\leq \sum_{t=1}^{T} \frac{1}{2\eta_t} \left[(z^t - z^*)^2 - (z^{t+1} - z^*)^2\right] + \frac{\eta_t}{2} G^2 \\ &\leq \frac{1}{2\eta_1}(z^1 - z^*)^2 - \frac{1}{2\eta_T}(z^{T+1} - z^*)^2 + \frac{1}{2}\sum_{t=2}^{T}(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}})(z^t - z^*)^2 + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t \\ &\leq D^2\left(\frac{1}{2\eta_1} + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right) + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t \\ &\leq \frac{D^2}{2\eta_T} + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t \end{aligned} \tag{3.8}$$

The third inequality holds since we assume $\{\eta_t\}$ is non-increasing. If define $\eta_t = 1/\sqrt{t}$, then

$$\begin{aligned} \sum_{t=1}^{T}\eta_t &= \sum_{t=1}^{T}\frac{1}{\sqrt{t}} \\ &\leq 1 + \int_{t=1}^{T}\frac{dt}{\sqrt{t}} \\ &\leq 2\sqrt{T} - 1 \end{aligned} \tag{3.9}$$

Substitute Equation (3.9) into Equation (3.8) yields the result. □

Alternatively, one can use a fixed step size that leads to an equivalent regret bound.

Corollary 1 (Regret bound, Ben-Tal et al. (2015b))**.** *If $\eta_t = D/(G\sqrt{T})$, the regret of OGD algorithm is*

$$R_{OGD}(T) \leq GD\sqrt{T}$$

Proof. $\eta_t = D/(G\sqrt{T})$ satisfies the non-increasing requirement, and hence Equation (3.8) holds. Meanwhile,

$$\sum_{t=1}^{T}\eta_t = T\frac{D}{G\sqrt{T}} = \frac{D\sqrt{T}}{G} \tag{3.10}$$

Substitute Equation (3.10) and $\eta_T = D/(G\sqrt{T})$ into Equation (3.8) yields the result. □

Corollary 1 reveals the convergence rate of the OGD algorithm is *sub-linear*. That is, it requires at least $\mathcal{O}(1/\epsilon^2)$ steps to make $R_{OGD} \leq \epsilon$.

3.3 Oracle-based Algorithm

In this section, we can formally describe the oracle-based approximate robust optimization algorithm and show that the solution is 2ϵ-approximate optimal.

An oracle, denoted by $\mathcal{O}_\epsilon(z)$ for $z \in \mathcal{Z}$, solves the nominal minimization problem in Equation (3.2) with known parameter vector z. It returns a new iterator x or correctly claim the problem is "infeasible".

Remark 1. *Ben-Tal et al. (2015b) proposed a feasible oracle and claimed that the optimality can be obtained via binary search. However, Ho-Nguyen & Kılınç-Karzan (2018) showed that the final output is not only ϵ-feasible but also ϵ-optimal if replace the feasible oracle with a minimization oracle. In this project, we implemented the minimization oracle. That is, with an updated parameter vector z, we solve the nominal problem with a convex optimizer.*

The oracle-based approximate robust optimization algorithm is presented in Algorithm 3.1. The algorithm is composed of two parts in each iteration. In the first part, a new uncertainty vector is obtained with the OGD algorithm, and in the second part, an ϵ-approximate feasible solution to the Equation (3.1) is obtained by calling the oracle.

Theorem 2 (Convergence, Ben-Tal et al. (2015b))**.** *With fixed step size $\eta_t = D/(G\sqrt{T})$, Algorithm 3.1 returns a 2ϵ-approximate feasible, ϵ-approximate optimal solution to Equation (3.1) or correctly concludes that it is infeasible. The algorithm terminates after at most $\mathcal{O}(1/\epsilon^2)$ calls to the oracle $\mathcal{O}_\epsilon(z)$.*

Proof. When Algorithm 3.1 returns "INFEASIBLE", by definition of the oracle, it can happen only when for some $t \leq T$ there exists no $x \in \mathcal{D}$ such that

$$f_i(x, z_i^t) \leq 0, \quad i = 1, \cdots, m$$

Since $z^t \in \mathcal{Z}$ is guaranteed, the above implies Equation (3.1) is infeasible.

Now, suppose a solution

$$\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x^t$$

is returned. Clearly $\bar{x} \in \mathcal{D}$. We know that each x^t is ϵ-approximate feasible to Equation (3.2), which implies

$$\forall\, i \in [m], \quad \frac{1}{T}\sum_{t=1}^{T} f_i(x^t, z^t) \leq \epsilon \tag{3.11}$$

On the other hand, from the regret bound of OGD algorithm we have

$$\max_{z_t \in \mathcal{Z}_i} \frac{1}{T}\sum_{t=1}^{T} f_i(x^t, z_i) - \frac{1}{T}\sum_{t=1}^{T} f_i(x^t, z_i^t) \leq \frac{GD}{\sqrt{T}} \leq \epsilon \tag{3.12}$$

Combining Equation (3.11) and Equation (3.12) we conclude that for all $i \in [m]$,

$$\epsilon \geq \frac{1}{T}\sum_{t=1}^{T} f_i(x^t, z_i^t) \geq \max_{z_t \in \mathcal{Z}_i} \frac{1}{T}\sum_{t=1}^{T} f_i(x^t, z_i) - \epsilon$$

which, along with the convexity of f_i in x, implies

$$\max_{z_i \in \mathcal{Z}_i} f_i(\bar{x}, z_i) \leq \max_{z_t \in \mathcal{Z}_i} \frac{1}{T}\sum_{t=1}^{T} f_i(x^t, z_i) \leq 2\epsilon$$

which shows $\bar{x}$ is 2ϵ-feasible to the original robust optimization problem. Furthermore, let's define

$$OPT^t := \min_x \{f_0(x) : f_i(x, z_i^t) \leq 0,\, i \in [m],\, x \in \mathcal{D}\}$$

and OPT the best worst-case objective value for Equation (3.1). We know that

$$f_0(x^t) \leq OPT^t + \epsilon$$

Since the solution to Equation (3.1) is feasible for the nominal problem at iteration t, hence

$$f_0(x^t) \leq OPT^t + \epsilon \leq OPT + \epsilon$$

Using the convexity of f_0 we have

$$f_0(\bar{x}) \leq \frac{1}{T}\sum_{t=1}^{T} f_0(x^t) \leq \frac{1}{T}\sum_{t=1}^{T} f_0(x^t) \leq OPT + \epsilon \tag{3.13}$$

which implies $\bar{x}$ is 2ϵ-optimal. □

Remark 2. *From implementation perspective, Algorithm 3.1 requires to obtain the diameter of the uncertainty set D and the upper bound of the norm of the gradient G. Both may not be easy to determine. Alternative, the dynamic step size in Theorem 1 is easy to calculate. Although the number of iterations T is not as straightforward, one can rely on early termination of the algorithm (please see Section 5.1). Hence, we will use dynamic step size when implementing Algorithm 3.1. Theorem 1 still hold.*

Algorithm 3.1 : Dual-subgradient algorithm for ϵ-approximate robust optimization (Ben-Tal et al., 2015a).

Input: target accuracy $\epsilon > 0$, parameter D and G

Initialize $z^0 = [z_i^0, \cdots, z_m^0]^T \in \mathcal{Z}$ arbitrarily, $T = \lceil G^2 D^2 \epsilon^{-2} \rceil$, and $\eta_t = D/(G\sqrt{T})$

Set $x^0 \leftarrow \mathcal{O}_\epsilon\left(z^0\right)$

Set $t = 1$

while $t < T$ **and** $|f(x^t) - f(x^{t-1})| > \epsilon$ **do**

  for $i = 1, \ldots, m$ **do**

    $z_i^t \leftarrow P\left(z_i^{t-1} + \eta_t \nabla_{z_i} f_i\left(x^{t-1}, z_i^{t-1}\right)\right)$

  end for

  $x^t \leftarrow \mathcal{O}_\epsilon\left(z^t\right)$.

  if Oracle declared infeasibility **then**

    return "INFEASIBLE"

  end if

  $t \leftarrow t + 1$

end while

return $\bar{x} = \frac{1}{T}\sum_{t}^{T} x^t$

4 Case Study

In this section, we describe the problem used to analyze the performance of the Approximate RO algorithm, as well as the different instances and uncertainty set configurations tested.

4.1 Problem Description

We tackle the *Advertisement Campaign with Uncertain Exposure Rate Problem* presented in Delage (2021), where a company needs to plan the placement of ads to promote a new product on several different websites defined by set I. For each website $i \in I$, $h_i(\cdot)$ defines the expected number of consumers who will buy the product after seeing ads on that website. Each website $i \in I$ charges a daily price p_i for each ad placement. Also, the company has a total daily budget of B for the advertisement campaign.

It is difficult to estimate the conversion function $h_i(\cdot)$ for each website. However, the company expects that the number of consumers converted per additional ad placement decreases as more ads are being displayed, being reasonable to consider the function to be non-decreasing, with the behaviour shown below, for some $c_i \geq 0$, $d_i \geq 0$, and $0 < a_i \leq 1$, $\forall i \in I$, where $x_i \in \mathbb{R}_+$ defines the total number of daily ad exposures on website i.

$$h_i(x_i) := c_i \cdot \left(1 + \frac{x_i}{d_i}\right)^{a_i} - c_i$$

The company wants to maximize product sales, i.e., maximize the number of customer conversions per ad placement. The nominal nonlinear programming (NLP) problem can then be described as shown in Equation (4.1).

$$\begin{aligned}
\underset{x \in \mathbb{R}_+}{\text{maximize}} \quad & \sum_{i \in I} h_i(x_i) && && \text{(4.1a)}\\
\text{subject to} \quad & \sum_{i \in I} p_i \cdot x_i \leq B && && \text{(4.1b)}\\
& x_i \geq 0 && \forall i \in I && \text{(4.1c)}
\end{aligned}$$

However, given the presence of uncertainty, a_i becomes an uncertain parameter defined as $a_i(z_i) := \bar{a}_i \cdot (1 - 0.25 z_i)$, $\forall i \in I$, given a random vector $z \in \mathcal{Z}$, with $\mathcal{Z}$ being the known uncertainty set. The problem can then be formulated as shown in Equation (4.2).

$$\begin{aligned}
\underset{x \in \mathbb{R}_+,\, t \in \mathbb{R}}{\text{maximize}} \quad & t && && \text{(4.2a)}\\
\text{subject to} \quad & t \leq \sum_{i \in I} \left[c_i \cdot \left(1 + \frac{x_i}{d_i}\right)^{a_i(z_i)} - c_i \right], && \forall a_i(z_i) \in \mathcal{U} && \text{(4.2b)}\\
& \sum_{i \in I} p_i \cdot x_i \leq B && && \text{(4.2c)}\\
& x_i \geq 0 && \forall i \in I && \text{(4.2d)}
\end{aligned}$$

Given this is a robust nonlinear programming problem, given *Theorem 6.2* in Delage (2021), given $v \in \mathbb{R}^n$, we can construct the Fenchel Robust Counterpart (FRC) (Ben-Tal et al., 2015a) with the help of the concave conjugate function g_*, defined as

$$g_*(x, v) := \inf_{z \in \mathcal{Z}_g} v^T z - g(x, z),$$

and the support function δ^*, defined as

$$\delta^*(v \mid \mathcal{Z}) := \sup_{z \in \mathcal{Z}} z^T v.$$

In Equation (4.2b) we have a sum of separable functions, and from Table A.1.2 we can construct the FRC for the problem as shown in Equation (4.3).

$$\underset{x,\, y,\, v,\, t}{\text{maximize}} \quad t \tag{4.3a}$$

$$\text{subject to} \quad t + \delta^*(v \mid \mathcal{U}) - \sum_{i \in I} \left[\frac{v_i}{\ln(y_i)} \ln\left(\frac{-v_i/c_i}{\ln(y_i)} \right) - \frac{v_i}{\ln(y_i)} \right] + \sum_{i \in I} c_i \leq 0, \tag{4.3b}$$

$$y_i = 1 + \frac{x_i}{d_i} \qquad \forall i \in I \tag{4.3c}$$

$$\sum_{i \in I} p_i \cdot x_i \leq B \tag{4.3d}$$

$$x_i \geq 0 \qquad \forall i \in I \tag{4.3e}$$

Next, we describe the uncertainty set configurations used, and the respective tractable reformulations of Equation (4.3) for the different support functions $\delta^*(v \mid \mathcal{U})$. The results from the reformulations shown in the following sections were then used for comparison with the Approximate RO algorithm results.

4.1.1 Budgeted Uncertainty Set

We first analyze the *Advertisement Campaign with Uncertain Exposure Rate Problem* in Equation (4.3) considering that the uncertainty set for z is a *Budgeted-like Uncertainty Set* $\mathcal{Z}_1$, such that:

$$\mathcal{Z}_1 := \left\{ z \in \mathbb{R}^n \;\middle|\; 0 \leq z \leq 1,\ \sum_{i \in I} z_i \leq \Gamma \right\}$$

and vector $a \in \mathcal{U}_1$, such that:

$$\mathcal{U}_1 := \{ a \in \mathbb{R}^n \mid \forall z \in \mathcal{Z}_1, a_i(z_i) = \bar{a}_i(1 - 0.25 z_i), \forall i \in I \}$$

with $\bar{a} \in [0,\ 1]^n$. For $\mathcal{U}_1$, the support function δ^* will then be defined as

$$\delta^*(v \mid \mathcal{U}_1) := \bar{a}^T v + \delta^*(-0.25 \mathbf{diag}(\bar{a}) v \mid \mathcal{Z}_1).$$

With the help of Table A.1.1, we can see that $\delta^*(v \mid \mathcal{Z}_1)$ will be

$$\delta^*(v \mid \mathcal{Z}_1) := \underset{\lambda \in \mathbb{R}, w \in \mathbb{R}^n : \lambda \geq v - w, \lambda \geq 0, w \geq 0}{min} \quad \sum_{i \in I} w_i + \lambda \Gamma,$$

and the tractable reformulation for the problem considering uncertainty set $\mathcal{Z}_1$ as shown in Equation (4.4).

$$\underset{t,\,x,\,y,\,v,\,\lambda,\,w}{\text{maximize}} \quad t \tag{4.4a}$$

$$\text{subject to} \quad t + \bar{a}^T v + \lambda\Gamma + \sum_{i\in I} w_i + \sum_{i\in I}\left[c_i - \left(\frac{v_i}{\ln(y_i)}\ln\left(\frac{-v_i/c_i}{\ln(y_i)}\right) - \frac{v_i}{\ln(y_i)}\right)\right] \leq 0, \tag{4.4b}$$

$$y_i = 1 + \frac{x_i}{d_i}, \qquad \forall i \in I \tag{4.4c}$$

$$\sum_{i\in I} p_i \cdot x_i \leq B \tag{4.4d}$$

$$\lambda \geq -0.25\mathbf{diag}(\bar{a})v - w \tag{4.4e}$$

$$x \geq 0 \tag{4.4f}$$

$$v \leq 0 \tag{4.4g}$$

$$w \geq 0 \tag{4.4h}$$

$$\lambda \geq 0 \tag{4.4i}$$

4.1.2 Ellipsoidal Uncertainty Set

Next, we analyze problem Equation (4.3) considering that the uncertainty set for z is an *Ellipsoidal Uncertainty Set* $\mathcal{Z}_2$, such that:

$$\mathcal{Z}_2 := \{z \in \mathbb{R}^n \mid ||z||_2 \leq \rho\}.$$

We then have $a \in \mathcal{U}_2$ and the support function δ^*, such that:

$$\mathcal{U}_2 := \{a \in \mathbb{R}^n \mid \forall z \in \mathcal{Z}_2, a_i(z_i) = \bar{a}_i(1 - 0.25z_i), \forall i \in I\}$$

$$\delta^*(v \mid \mathcal{U}_2) := \bar{a}^T v + \delta^*(-0.25\mathbf{diag}(\bar{a})v \mid \mathcal{Z}_2).$$

With the help of Table A.1.1, we can see that $\delta^*(v \mid \mathcal{Z}_2)$ will be defined as

$$\delta^*(v \mid \mathcal{Z}_2) := \rho||v||_2,$$

and the tractable reformulation for the problem considering the ellipsoidal uncertainty set $\mathcal{Z}_2$ as shown in Equation (4.5).

$$\underset{t,\,x,\,y,\,v}{\text{maximize}} \quad t \tag{4.5a}$$

$$\text{subject to} \quad t + \bar{a}^T v + s + \sum_{i\in I}\left[c_i - \left(\frac{v_i}{\ln(y_i)}\ln\left(\frac{-v_i/c_i}{\ln(y_i)}\right) - \frac{v_i}{\ln(y_i)}\right)\right] \leq 0, \tag{4.5b}$$

$$y_i = 1 + \frac{x_i}{d_i}, \qquad \forall i \in I \tag{4.5c}$$

$$\sum_{i\in I} p_i \cdot x_i \leq B \tag{4.5d}$$

$$s \geq \rho \cdot ||-0.25\mathbf{diag}(\bar{a})v||_2 \tag{4.5e}$$

$$x \geq 0 \tag{4.5f}$$

$$v \leq 0 \tag{4.5g}$$

4.1.3 CUSTOMIZED UNCERTAINTY SET

Lastly, we analyze problem Equation (4.3) considering that the uncertainty set for z is a *Customized Uncertainty Set* $\mathcal{Z}_3$, such that:

$$\mathcal{Z}_3 := \left\{ z \in \mathbb{R}^n \;\middle|\; z \geq 0,\ \sum_i z_i = 1,\ \sum_i z_i \ln(z_i) \leq \rho \right\},$$

for which $a \in \mathcal{U}_3$, and $\mathcal{U}_3$ and δ^* are defined as:

$$\mathcal{U}_3 := \{a \in \mathbb{R}^n \mid \forall z \in \mathcal{Z}_3, a_i(z_i) = \bar{a}_i(1 - 0.25 z_i), \forall i \in I\}$$

$$\delta^*(v \mid \mathcal{U}_3) := \bar{a}^T v + \delta^*(-0.25\mathbf{diag}(\bar{a})v \mid \mathcal{Z}_3).$$

We can see that $\mathcal{Z}_3$ is the intersection of a KL-Divergence uncertainty set $\mathcal{Z}_3^1$ and the uncertainty set $\mathcal{Z}_3^2 := \left\{ z \in \mathbb{R}^n \;\middle|\; z \geq 0,\ \sum_{i \in I} z_1 = 1 \right\}$. With the help of Table A.1.1 and *Theorem 6.6* in Delage (2021), $\delta^*(v \mid \mathcal{Z}_3)$ will be:

$$\delta^*(v \mid \mathcal{Z}_3) := \min_{\lambda, \mu \geq 0, w^1, w^2 : \lambda \geq w^1, w^1 + w^2 = v} \lambda + \sum_{i \in I} \mu \cdot \exp\left(\frac{w_i^2}{\mu} - 1\right),$$

and the tractable reformulation for the problem considering uncertainty set $\mathcal{Z}_3$ as shown in Equation (4.6).

$$\begin{aligned}
&\underset{t,\, x,\, y,\, v,\, \lambda,\, w,\, \mu,\, s}{\text{maximize}} && t && && \text{(4.6a)}\\
&\text{subject to} && t + \bar{a}^T v + s + \sum_{i \in I} \left[c_i - \left(\frac{v_i}{\ln(y_i)} \ln\left(\frac{-v_i / c_i}{\ln(y_i)} \right) - \frac{v_i}{\ln(y_i)} \right) \right] \leq 0, && && \text{(4.6b)}\\
& && y_i = 1 + \frac{x_i}{d_i}, && \forall i \in I && \text{(4.6c)}\\
& && \sum_{i \in I} p_i \cdot x_i \leq B && && \text{(4.6d)}\\
& && s \geq \lambda + \sum_{i \in I} \mu \cdot \exp\left(\frac{w_i}{\mu} - 1 \right) + \rho\mu && && \text{(4.6e)}\\
& && \lambda \geq -0.25\mathbf{diag}(\bar{a})v - w && && \text{(4.6f)}\\
& && x \geq 0 && && \text{(4.6g)}\\
& && v \leq 0 && && \text{(4.6h)}\\
& && \mu \geq 0 && && \text{(4.6i)}
\end{aligned}$$

4.2 PROBLEM INSTANCES

For the study conducted, we used the different problem instances described next.

4.2.1 BASE CASE

The base case used to verify the implementation using SciPy `optimize` functions[2] and the convergence of the algorithm 3.1 uses the problem instance with the number of websites $n = 4$ and the budgeted uncertainty set from Section 4.1.1, with the values for the parameters being:

[2] `https://docs.scipy.org/doc/scipy/reference/optimize.html`

- $\bar{a}^T = [0.2000,\ 0.1875,\ 0.1625,\ 0.1500]$
- $c^T = [30.0,\ 30.0,\ 30.0,\ 30.0]$
- $d^T = [1000.0,\ 1000.0,\ 1000.0,\ 1000.0]$
- $p^T = [0.110,\ 0.085,\ 0.090,\ 0.080]$
- $B = 1.0$

The base case with the values above is also used with the ellipsoidal and customized uncertainty sets from Section 4.1.2 and Section 4.1.3.

4.2.2 Scaled Problems

To analyze the performance of the Approximate RO algorithm for large-scale problems, we increase the size of the advertisement campaign problem by increasing the number n of available websites. The values for the parameters are then defined as follows:

- $\bar{a}_i$ are randomly sampled from an uniform distribution within the interval $[0.15,\ 0.20]$, i.e., $\bar{a} \sim U(0.15,\ 0.20)^n$.
- p_i are randomly sampled from an uniform distribution within the interval $[0.80,\ 1.10]$ and then multiplied by 0.1, i.e., $p \sim U(0.080,\ 0.110)^n$.
- $c_i = 30.0$ remains the same $\forall i \in I$
- $d_i = 1000.0$ remains the same $\forall i \in I$
- $B = 1.0$ remains unchanged.

Starting from the base case with $n = 4$, we increase the number of available websites until $n = 20$. To maintain reproducibility, for a given instance n, the same seed value is used to instantiate the random number generator that samples values for $\bar{a}_i$ and p_i.

5 Numerical Analysis

In the present work, we investigate the practical value of the Dual-Subgradient algorithm for Approximate RO by comparing its results to the solutions to the RC reformulation presented in Section 4.1.1 to Section 4.1.3 and test the algorithm's rate of convergence and running time. In this section, we present and discuss the results achieved for the different tests conducted.

5.1 Algorithm Convergence

As stated in Ben-Tal et al. (2015b), the number of iterations depends only on the size of the uncertainty set, the speed of change of the constraints concerning the uncertain parameters, and the approximation criteria, with algorithm 3.1 returning a 2ϵ-approximate solution to the original RC formulation or correctly concluding that the problem is infeasible. The algorithm's worst-case time complexity then being $O(1/\epsilon^2)$, i.e., the algorithm terminates after at most $O(1/\epsilon^2)$ calls of the oracle $\mathcal{O}_\epsilon$.

We then start by testing the rate of convergence for the Dual-Subgradient algorithm. By choosing the approximation accuracy $\epsilon = 10^{-6}$, the above time complexity worst-case bound implies that the algorithm would terminate after 10^{12} iterations. But are these many iterations required?

Using the base case problem (Section 4.2.1) with a budgeted uncertainty set (Section 4.1.1) and approximation accuracy $\epsilon = 10^{-6}$, we test the algorithm's objective function value convergence by gradually increasing the maximum number of iterations T from 1 to 1000.

The results are plotted in Figure 5.1, where we see that the objective function value converges at a sublinear rate and is sufficiently close to 0.0538 after approximately 140 iterations, much lower than the worst-case bound. This behaviour was noticed throughout the tests performed and points to the algorithm's applicability in a small finite number of iterations for an ϵ-approximate solution.

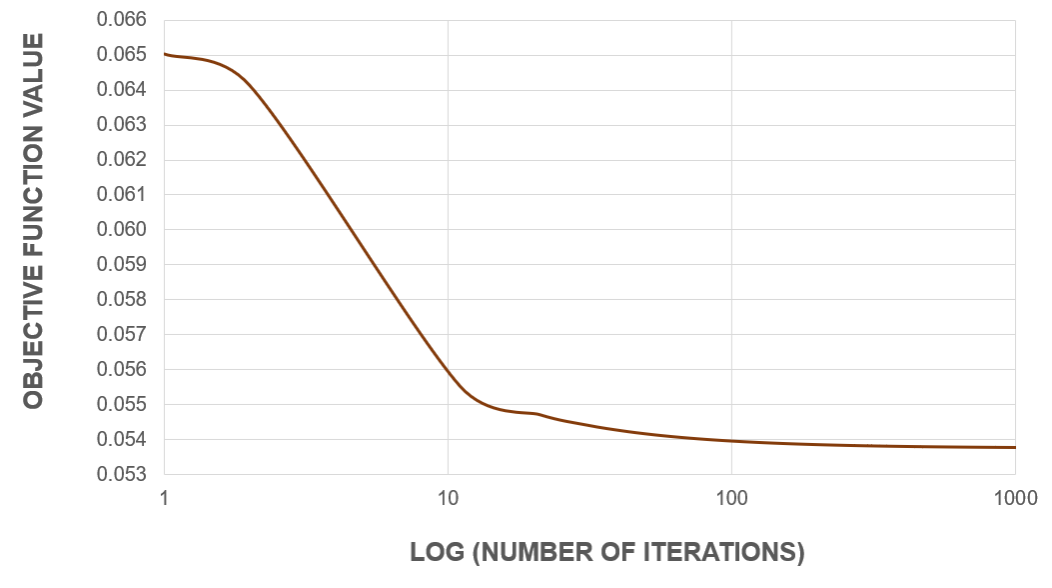

Figure 5.1: Convergence of objective function value over the number of iterations for the base case problem with Budgeted uncertainty set.

5.2 Verification of RC Reformulation and Base Case Results

The RC reformulation from Section 4.1.1 is provided in Delage (2021) with a MATLAB® code implementation using YALMIP[3]. However, since all the work presented herein is implemented using Python 3 API (Van Rossum & Drake, 2009) using SciPy `optimize` functions, we start with a numerical verification between the MATLAB-YALMIP and SciPy implementations to ensure the Python code is correct. Table 5.2.1 shows the values achieved with both implementations, from which it is possible to verify the correctness of the SciPy solution.

Table 5.2.1: Code verification using base case problem (Section 4.2.1) with $\Gamma = 1$ and $\epsilon = 10^{-6}$.

Solver	**Objective Function Value**	**Solution Vector** (x^T)
MATLAB-YALMIP	0.0538	$(2.391,\ 2.551,\ 2.944,\ 3.190)$
SciPy (`trust-constr`)	0.0538	$(2.391,\ 2.552,\ 2.943,\ 3.190)$
ϵ-Approximate RO Algo.	0.0538	$(2.671,\ 2.845,\ 2.637,\ 2.838)$

Next, we solve the base case problem with the Dual-Subgradient algorithm, using an approximation accuracy $\epsilon = 10^{-6}$, and compare the results with the ones from the RC reformulation to verify the solution performance of the ϵ-Approximate RO algorithm (Table 5.2.1). We see that the algorithm can achieve the same objective function value as the RC reformulation, albeit with a different solution x. However, since the case study is an NLP, we can have multiple optimal solutions to the problem achieving the same optimal objective function value, hence the solutions x achieved by different methods need not be identical. Therefore, the results from the Dual-Subgradient algorithm show its correctness and applicability.

To analyze the solutions from the RC reformulation and the ϵ-Approximate RO algorithm (Table 5.2.1) when confronted with uncertainty, we test their performance against 1000 realizations

[3] `https://yalmip.github.io`

of the random vector $z \in \mathcal{Z}_1$. Table 5.2.2 displays the results for this test, where it is possible to notice that, while the ϵ-Approximate RO algorithm is on average slightly more optimistic than the exact model, both methods achieve close mean objective function values and worst-case scenario values, once more pointing to the algorithm's applicability to robust optimization NLP problems.

Table 5.2.2: Performance of solutions from RC reformulation and ϵ-Approximate RO algorithm under 1000 realizations $z \in \mathcal{Z}_1$.

Solution Method	**Average Value**	**Standard Deviation**	**Min / Max Values**
NLP RC reformulation	0.0540	3.2423×10^{-4}	0.0538 / 0.0557
ϵ-Approximate RO Algo.	0.0542	4.0853×10^{-4}	0.0537 / 0.0567

5.3 Testing Different Uncertainty Sets

Next, we compare the results for the NLP RC reformulations with Ellipsoidal (Section 4.1.2) and Customized (Section 4.1.3) uncertainty sets to the results from the ϵ-Approximate RO algorithm to test its generalization. We utilize the base case problem configuration (Section 4.2.1) with the previously described uncertainty sets, both with parameter $\rho = 1.0$, and the approximation accuracy $\epsilon = 10^{-6}$. After the problem is solved using both methods, we test the two solutions against 1000 realizations of the random vectors $z \in \mathcal{Z}_2$, for the Ellipsoidal uncertainty set, and $z \in \mathcal{Z}_3$, for the Customized uncertainty set, and report the worst-case objective function values for the two methods.

Table 5.3.1: Results for the base case problem (Section 4.2.1) with Ellipsoidal uncertainty set ($\rho = 1.0$) and $\epsilon = 10^{-6}$.

Solution Method	**Worst-Case Value**	**Solution Vector** (x^T)
NLP RC reformulation	0.0514	$(1.414,\ 5.707,\ 1.528,\ 2.774)$
ϵ-Approximate RO Algo.	0.0514	$(1.476,\ 6.187,\ 1.440,\ 2.277)$

Table 5.3.1 and Table 5.3.2 show that the ϵ-Approximate RO algorithm is able to reach the same worst-case objective function value for the Ellipsoidal uncertainty set and a very close worst-case objective function value for the Customized uncertainty set, respectively. Therefore, for a given problem, we see that the algorithm's performance is not impacted by the uncertainty set configuration, an attribute that favours its application over the NLP RC when the problem's reformulation proves to be cumbersome given an elaborated uncertainty set, such as the case for the Customized uncertainty set $\mathcal{Z}_3$.

Table 5.3.2: Results for the base case problem (Section 4.2.1) with Customized uncertainty set ($\rho = 1.0$) and $\epsilon = 10^{-6}$.

Solution Method	**Worst-Case Value**	**Solution Vector** (x^T)
NLP RC reformulation	0.0537	$(2.425,\ 2.689,\ 2.657,\ 3.318)$
ϵ-Approximate RO Algo.	0.0536	$(2.696,\ 2.997,\ 2.653,\ 2.623)$

5.4 ALGORITHM RUNNING TIME

Next, we analyze the CPU run time for the ϵ-Approximation RO algorithm as the problem size increases. The different problem instances are constructed as described in Section 4.2.2. For each instance n, 50 different problem configurations are constructed and solved with the algorithm, from which we report the average number of iterations and average CPU run time (in seconds) the algorithm required to solve the problem. We repeated this test for the three different uncertainty sets described in Sections 4.1.1, 4.1.2 and 4.1.3.

Table 5.4.1 shows the results for the test, from which it is clear that the algorithm remains attractive even for larger problem instances, with the average number of iterations not exceeding 217 and the average running time increasing by a factor smaller than the increase in the problem size. For example, using the Budgeted uncertainty set, the problem with $n = 5$ requires on average 144 iterations and 34.07 seconds to be solved, while the problem with $n = 20$ requires on average 189 iterations and 129.19 seconds to be solved, which is $0.3\times$ more iterations and $2.8\times$ more CPU run time for solving a problem that is $4\times$ larger than the former.

Table 5.4.1: Average CPU run time and number of iterations for different problem instances with Budgeted ($\Gamma = 1$), Ellipsoidal ($\rho = 1.0$) and Customized ($\rho = 1.0$) uncertainty sets.

Problem Instance	**Average Number Iterations**			**Average Run Time** (sec.)		
	Budgeted	*Ellipsoidal*	*Customized*	*Budgeted*	*Ellipsoidal*	*Customized*
4	176	138	197	29.83	29.97	33.25
5	144	158	143	34.07	48.50	29.92
6	209	173	186	49.83	60.14	46.48
8	169	184	189	56.92	107.88	63.28
10	152	194	188	60.53	128.30	67.04
20	189	-	217	129.19	-[4]	148.86

In this test, we do not compare the solutions from the NLP RC reformulations with the ones from the algorithm since for all the instances where $n > 4$, the NLP RC reformulations exceeded the maximum of 10000 iterations to solve the problem without reaching the optimal value. This proved to be even more challenging for the Budgeted and Customized sets, for which the maximum of 10000 iterations was exceeded for several instance configurations with $n = 4$. This shows the ϵ-Approximate RO algorithm to be extremely attractive for larger instance sizes, providing that the algorithm is robust when confronted with problem scalability.

However, we have to pointed out that the expected total run time increases linearly to the dimension of the problem despite the number of iterations does not change much. This shows strong evidence that the algorithm complexity (number of iterations) does not depend on the dimension of the problem. However, the increasing run time is attributed to the fact that the oracle complexity increases with the dimension of the problem, which can be a main drawback of this algorithm to a very large scaled problem.

[4] Value not reported due to calculation error (such as division by zero or NaN) by solver during run time.

6 CONCLUSIONS

We have implemented an oracle-based approximate algorithm to solve a convex-concave robust optimization problem and shown that, upon applying a ϵ-optimal oracle, the algorithm yields a 2ϵ-approximate feasible, ϵ-approximate optimal solution.

We have proved sublinear convergence rate both in theory and with numerical study. Next, we demonstrate that this approximate framework is easier to implement than the reformulation via Fenchel duality. Moreover, by comparing both approximate and classical approaches in the Ad Campaign problem, we have shown that the approximate algorithm is accurate, reliable and outperforms the classical approach in handling even moderate scale problems. From the results achieved, we highly recommend this approximate approach in solving large-scale convex-concave robust optimization due of its easy implementation and reliable convergence performance

As pointed out the in numerical investigations, the complexity of using the oracle is linearly dependent on the dimension of the problem, which limits the application of the algorithm in larger scaled problems. One direction to extend this work is to replacing the oracle with some other online learning technique. Indeed, as pointed in Ho-Nguyen & Kılınç-Karzan (2018), a purely online first-order update based algorithm was able to achieve that.

CODE AVAILABILITY

All the code and scripts were implemented using Python 3 API (Van Rossum & Drake, 2009) and are available under MIT License. The optimization models were implemented using SciPy (Virtanen et al., 2020) and NumPy (Harris et al., 2020) modules and were designed to be easily modified and facilitate tests. The code was also implemented to run in any system, avoiding OS-specific commands and packages.

All codes are available in the repository on Mr. Quan Zhou's Github webpage at the address: `https://github.com/qzhou-2020/HEC_MATH_80624_project`.

Alternatively, the code can also be found in the repository on Mr. Luiz Resende Silva's GitHub webpage at the following address (forked from the former repository): `https://github.com/luiz-resende/HEC_MATH_80624_project`.

STATEMENT OF CONTRIBUTIONS

We at this moment state that all the work presented in this report and the code files submitted is that of the authors and that any third-party material was used according to the project's instructions, being duly referenced and following any applicable copyright laws.

REFERENCES

Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization – methodology and applications. *Mathematical Programming*, 92(3):453–480, 1 May 2002. doi: 10.1007/s101070100286. URL `https://doi.org/10.1007/s101070100286`.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009. URL `https://www2.isye.gatech.edu/~nemirovs/FullBookDec11.pdf`.

Aharon Ben-Tal, Dick den Hertog, and Jean-Philippe Vial. Deriving robust counterparts of nonlinear uncertain inequalities. *Mathematical Programming*, 149:265–299, Feb 2015a. doi: 10.1007/s10107-014-0750-8. URL `https://doi.org/10.1007/s10107-014-0750-8`.

Aharon Ben-Tal, Elad Hazan, Tomer Koren, and Shie Mannor. Oracle-based robust optimization via online learning. *Operations Research*, 63(3):628–638, May 2015b. doi: 10.1287/opre.2015.1374. URL `https://doi.org/10.1287/opre.2015.1374`.

Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011. ISSN 00361445. doi: 10.2307/23070141. URL http://www.jstor.org/stable/23070141.

Giuseppe Calafiore and M. C. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102:25–46, Jan 2005. ISSN 1436-4646. doi: 10.1007/s10107-003-0499-y. URL https://doi.org/10.1007/s10107-003-0499-y.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006. doi: 10.1017/CBO9780511546921. URL https://doi.org/10.1017/CBO9780511546921.

Erick Delage. *Quantitative Risk Management Using Robust Optimization - Lecture Notes*. Department of Decision Sciences. HEC Montréal, 12 Dec 2021. URL https://tintin.hec.ca/pages/erick.delage/MATH80624_LectureNotes.pdf.

Bram L Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, 2015. ISSN 0305-0483. doi: 10.1016/j.omega.2014.12.006. URL https://www.sciencedirect.com/science/article/pii/S0305048314001698.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sep 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Nam Ho-Nguyen and Fatma Kılınç-Karzan. Online first-order framework for robust convex optimization. *Operations Research*, 66(6):1670–1692, 2018. doi: 10.1287/opre.2018.1764. URL https://doi.org/10.1287/opre.2018.1764.

Almir Mutapcic and Stephen Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods and Software*, 24(3):381–406, 2009. doi: 10.1080/10556780802712889. URL https://doi.org/10.1080/10556780802712889.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697. URL https://www.python.org.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2. URL https://rdcu.be/b08Wh.

Bo Zeng and Long Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, 2013.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 928–935. AAAI Press, 2003. ISBN 1577351894. URL https://www.aaai.org/Papers/ICML/2003/ICML03-120.pdf.

A Appendix

A.1 Reference tables for support function and partial concave conjugate

The following tables were used as reference for the construction of robust counterpart of nonlinear problem (Ben-Tal et al., 2015a).

Table A.1.1: Table of reformulations for uncertainty sets.

Uncertainty region	$\mathcal{Z}$	*Support function* $\delta^*(v\|\mathcal{Z})$
Box	$\|\|z\|\|_\infty \leq \rho$	$\rho\|\|v\|\|_1$
Ellipsoid	$\|\|z\|\|_2 \leq \rho$	$\rho\|\|v\|\|_2$
Polyhedral	$b - Bz \geq 0$	$\inf_{w\geq 0: B^T w = v} b^T w$
Cone	$b - Bz \in C$	$\inf_{w\in C^*: B^T w = v} b^T w$
KL-Divergence	$\sum_l z_l \log(\frac{z_l}{z_l^0}) \leq \rho$	$\inf_{u\geq 0} \sum_l z_l^0 u e^{(v_l/u)-1} + \rho u$
Geometric progression	$\sum_i \alpha_i e^{(d_i)^T z} \leq \rho$	$\inf_{u,w\geq 0: \sum_i d_i w_i = v} \sum_i \{w_i \log(\frac{w_i}{\alpha_i u}) - w_i\} + \rho u$
Intersection	$\mathcal{Z} = \cap_i \mathcal{Z}_i$	$\inf_{\{w_i\}: \sum_i w_i = v} \sum_i \delta^*(w^i\|\mathcal{Z}_i)$
Example	$\mathcal{Z}_k = \{z \mid \|\|z\|\|_k \leq \rho_k\}$ $k = 1, 2$	$\inf_{\{w^1, w^2\}: w^1 + w^2 = v} \rho_1 \|\|w^1\|\|_\infty + \rho_2 \|\|w^2\|\|_2$
Minkowski sum	$\mathcal{Z} = \mathcal{Z}_1 + \cdots + \mathcal{Z}_K$	$\sum_i \delta^*(v\|\mathcal{Z}_i)$
Convex hull	$\mathcal{Z} = \text{conv}(\mathcal{Z}_1, \ldots, \mathcal{Z}_K)$	$\max_i \delta^*(v\|\mathcal{Z}_i)$

Table A.1.2: Table of reformulations for constraint functions.

Constraint function	$g(x, z)$	*Partial concave conjugate* $g_*(x, v)$
Linear in z	$z^T g(x)$	$\begin{cases} 0 & \text{, if } v = g(x) \\ -\infty & \text{, otherwise} \end{cases}$
Concave in z, separable in z and x	$g(z)^T x$	$\sup_{\{s^i\}: \sum_{i=1}^n s^i = v} \sum_i x_i (g_i)_* \left(\frac{s^i}{x_i}\right)$
Example	$-\sum_i \frac{1}{2}(z^T Q_i z) x_i$	$\sup_{\{s^i\}: \sum_{i=1}^n s^i = v} -\frac{1}{2} \sum_{i=1}^n \frac{(s^i)^T Q_i^{-1} s^i}{x_i}$
Sum of functions	$\sum_i g_i(x, z)$	$\sup_{\{s^i\}: \sum_i s^i = v} \sum_i (g_i)_*(s^i, x)$
Sum of separable functions	$\sum_i g_i(x, z_i)$	$\sum_{i=1}^n (g_i)_*(v_i, x)$
Example	$-\sum_{i=1}^m x_i^{z_i}$, $x > 1, 0 \leq z \leq 1$	$\begin{cases} \sum_{i=1}^m \left(\frac{v_i}{\ln(x_i)} \ln\left(\frac{-v_i}{\ln(x_i)}\right) - \frac{v_i}{\ln(x_i)}\right) & \text{, if } v \leq 0 \\ -\infty & \text{, otherwise} \end{cases}$