A technical report on the paper “A learning-based algorithm to quickly compute good primal solutions for Stochastic Integer Programs” [2]

Warley Almeida Silva (20175894)

Abstract

Two-stage stochastic integer programs combine the hardness of solving stochastic programs with the hardness of solving integer programs. Given the theoretical results about the complexity of solving these programs, there has been continuous interest in the literature to propose novel algorithmic approaches that work well in practice. In this sense, the authors in [2] propose a learning-based algorithm to compute near-optimal solutions for two-stage stochastic integer programs, namely the representative scenario (RS) algorithm. This project aims to explain the RS algorithm, validate its performance, and compare it to other computational approaches. Even though the RS algorithm has shallow theoretical guarantees, computational experiments show that the RS algorithm manages to find near-optimal first-stage decisions while taking considerably less time than solving the original two-stage stochastic integer programs.

1 Introduction

Two-stage stochastic continuous programs with n first-stage decision variables, p first-stage constraints, m second-stage decision variables, and q second-stage constraints can be defined generally as $\tilde{P}$, where $A \in \mathbb{R}^{p\times n}$, $b \in \mathbb{R}^p$, $c \in \mathbb{R}^n$, $h(\xi) \in \mathbb{R}^q$, $q(\xi) \in \mathbb{R}^m$, $T(\xi) \in \mathbb{R}^{q\times n}$, $W \in \mathbb{R}^{q\times m}$, $x \in \mathbb{R}^n$, and $y \in \mathbb{R}^m$:

$$
\begin{aligned}
\tilde{P}: \quad & \min_{x\in\mathbb{R}^n} && c^T x + \mathbb{E}_\xi[\tilde{Q}(x,\xi)] && \text{(1a)}\\
& \text{subject to:} && Ax \le b && \text{(1b)}\\
\text{where } \tilde{Q}(x,\xi): \quad & \min_{y(\xi)\in\mathbb{R}^m} && q(\xi)^T y(\xi) && \text{(1c)}\\
& \text{subject to:} && Wy(\xi) \le h(\xi) - T(\xi)x && \text{(1d)}\\
& && y(\xi) \ge 0 && \text{(1e)}
\end{aligned}
$$

Two-stage stochastic continuous programs have been widely studied in the literature [3]. These programs can be difficult to solve because of two main reasons. First, the support set Ξ of the random variable ξ may not be discrete, so it becomes necessary to apply sampling techniques to sufficiently approximate it. Second, even when the support set Ξ is discrete, the number of realizations of the random variable ξ may be too large for some instances, which generates incredibly large programs. It is possible to solve $\tilde{P}$ by explicitly enumerating the realizations of the random variable ξ, thus generating a deterministic model, or implicitly exploring the solution space with the L-shaped algorithm [3].

Even though program $\tilde{P}$ describes a large number real-world applications, some applications additionally require the integrality of some decision variables in the first- and second-stage programs. However, the addition of integrality constraints considerably increases the difficulty of program $\tilde{P}$ because general integer programs are known to be NP-hard [1]. Thus, in addition to the difficulties of solving stochastic programs, two-stage stochastic integer programs encode undesirable properties of an mixed-integer objective function, which can be non-convex and even discontinuous [1], among other technical difficulties.

Two-stage stochastic integer programs with n first-stage decision variables, p first-stage constraints, m second-stage decision variables, and q second-stage constraints can be defined generally as P, where $A \in \mathbb{R}^{p\times n}$, $b \in \mathbb{R}^p$, $c \in \mathbb{R}^n$, $h(\xi) \in \mathbb{R}^q$, $q(\xi) \in \mathbb{R}^m$, $T(\xi) \in \mathbb{R}^{q\times n}$, $W \in \mathbb{R}^{q\times m}$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, and $\mathcal{I}_1$ and $\mathcal{I}_2$ are the sets of integer first- and second-stage variables, respectively:

$$
\begin{aligned}
P: \quad & \min_{x\in\mathbb{R}^n} && c^T x + \mathbb{E}_\xi[Q(x,\xi)] && \text{(2a)}\\
& \text{subject to:} && Ax \le b && \text{(2b)}\\
& && x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}_1 && \text{(2c)}\\
\text{where} \quad Q(x,\xi): \quad & \min_{y(\xi)\in\mathbb{R}^m} && q(\xi)^T y(\xi) && \text{(2d)}\\
& \text{subject to:} && Wy(\xi) \le h(\xi) - T(\xi)x && \text{(2e)}\\
& && y_i(\xi) \in \mathbb{Z} \quad \forall i \in \mathcal{I}_2 && \text{(2f)}
\end{aligned}
$$

Intuitively, one of the technical difficulties of solving program P is that, unlike its continuous sibling program $\tilde{P}$, solving the integer program $Q(\bar{x},\xi)$ in program P for a certain first-stage decision $\bar{x}$ is a difficult task, whereas solving the linear program $\tilde{Q}(\bar{x},\xi)$ for a certain first-stage decision $\bar{x}$ in program $\tilde{P}$ is an easy task. The hardness of solving two-stage stochastic integer programs, which inherently implies that likely there does not exist polynomial time algorithms to solve these programs efficiently, justifies the research towards algorithms that find a near-optimal solution within feasible computational efforts. In this sense, there have been efforts in the literature to adapt the L-shaped algorithm for general two-stage stochastic integer programs [6], as well as propose problem-specific improvements towards reaching a faster near-optimal solution [4].

The authors in [2] propose yet another approach to solve two-stage stochastic integer programs through the use of machine learning techniques. First, they outline the characteristics of the two-stage stochastic integer programs they will consider by stating two important assumptions about program P:

1. **Program P has complete recourse.** This assumption means that the second-stage problem is always feasible, and implies that for every feasible first-stage decision x there is at least one feasible second-stage decision $y(\xi)$ per feasible realization of the random variable $\xi \in \Xi$.

2. **Program P has finite support.** This assumption means that it is possible to enumerate the feasible realizations of the random variable ξ, i.e., the support set Ξ of the random variable ξ is a finite set.

The authors in [2] conjecture about the existence of a representative scenario $\xi^* = (q^*, T^*, h^*)$ that could well enough describe the support set Ξ. In this sense, one could solve the surrogate program S with the representative scenario $\xi^* =$

(q^*, T^*, h^*) instead of solving the deterministic program D, and still find the same optimal first-stage decision x. Note that program D is the deterministic version of program P built upon the assumption that the support set Ξ is finite. Based on this conjecture, the main idea of the authors in [2] is to avoid solving the larger problem D by guessing a representative scenario $\xi^* = (q^*, T^*, h^*)$ according to the instance $p = (A, b, c, W, \Xi)$, solving the smaller program S, and outputting its first-stage decision x as optimal to program P. Programs D and S have the following form with the previously defined components:

$$
\begin{aligned}
D: \quad \min_{x\in\mathbb{R}^n, y(\xi)\in\mathbb{R}^m} \quad & c^T x + \frac{1}{|\Xi|}\sum_{\xi\in\Xi} q(\xi)^T y(\xi) && \text{(3a)}\\
\text{subject to:} \quad & Ax \le b && \text{(3b)}\\
& x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}_1 && \text{(3c)}\\
& Wy(\xi) \le h(\xi) - T(\xi)x \quad \forall \xi \in \Xi && \text{(3d)}\\
& y(\xi)_i \in \mathbb{Z} \quad \forall \xi \in \Xi, \forall i \in \mathcal{I}_2 && \text{(3e)}
\end{aligned}
$$

$$
\begin{aligned}
S: \quad \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^m} \quad & c^T x + {q^*}^T y && \text{(4a)}\\
\text{subject to:} \quad & Ax \le b && \text{(4b)}\\
& x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}_1 && \text{(4c)}\\
& Wy \le h^* - T^* x && \text{(4d)}\\
& y_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}_2 && \text{(4e)}
\end{aligned}
$$

The representative scenario ξ^* does not have to be a part of the support set Ξ, i.e., it can be a completely artificial realization of the random variable ξ. However, solving program S with a representative scenario ξ^* does not generate an invalid first-stage decision for program P. Constraints (4b)-(4c) in program S guarantee first-stage feasibility for program P, whereas the complete recourse property guarantees second-stage feasibility for program P. Thus, a first-stage decision $\bar{x}$ obtained from program S is always a feasible solution to program P. The sole important thing is to guarantee that the representative scenario ξ^* guessed for a certain instance $p = (A, b, c, W, \Xi)$ sufficiently describes the support set Ξ, i.e., the objective values of programs P and S with a representative scenario ξ^* are either equal or sufficiently close. The authors in [2] use machine learning techniques, namely linear regression and artificial neural networks, to guess a representative scenario ξ^* for a certain instance p.

In this sense, this project aims to explain the representative scenario (RS) algorithm [2], implement it in a Jupyter notebook attached to this project, validate its ability to successfully find near-optimal first-stage decisions to program P, check its computational results presented by the authors in [2], and compare it with some other intuitive approaches to succinctly describe the support set Ξ. More specifically, this project studies the linear regression (LR) approach to predict a representative scenario ξ^* within the RS algorithm.

This project has the following structure. Section 2 explains how the RS algorithm works and how the linear regression model, which plays an important role within the algorithm, can be built. Section 3 presents the stochastic capacitated facility location problem, which has been chosen to study the performance of the algorithm, and explains additional aspects of the implementation of the RS algorithm for this problem. Section 4 discusses the computational performance of the RS algorithm with the LR approach in comparison to (i) the optimal solution of the deterministic model given by Gurobi and (ii) other solutions given by other intuitive approaches to succinctly describe the support set Ξ. Lastly, section 5 concludes this project with some remarks on the advantages and disadvantages of the RS algorithm, as well as some future work directions.

2 Methodology

This section explains how the RS algorithm works and how the linear regression model, which plays an important role within the algorithm, can be built.

2.1 Algorithm overview

First, let us formalize how the RS algorithm works:

Input: Instance information $p = (A, b, c, W, \Xi)$ and linear regression model;

Step 1 Guess representative scenario $\xi^* = (q^*, T^*, h^*)$ based on the instance information $p = (A, b, c, W, \Xi)$ with the linear regression model;

Step 2 Solve program S with the representative scenario $\xi^* = (q^*, T^*, h^*)$;

Step 3 Return the solution from program S as the solution of program P;

Output: The first-stage decision x obtained from solving program S.

The algorithm has a rather intuitive structure. The quality of the outputted first-stage decision x depends on the capacity of the linear regression model to guess good representative scenarios ξ^*. However, the generation of the linear regression model has two main technical difficulties. The first one is to generate training data for the linear regression model, i.e., a set of k pairs $(p_1, \xi_1^*), (p_2, \xi_2^*), ...(p_k, \xi_k^*)$ to train the linear regression model. Unlike other machine learning problems where input-output pairs can be trivially found, it is hard to find a representative scenario ξ_k for an instance p_k. Section 2.2 sheds light on how the input-output pairs can be generated for the training procedure.

The second one is to define exactly which parts of the instance information $p = (A, b, c, W, \Xi)$ are in fact important to guess the representative scenario $\xi^* = (q^*, T^*, h^*)$, thus determining what information will be passed onto the linear regression model. For example, the coefficient matrix A may be the same or sufficiently similar for all instances of a certain problem, so it will not play an important role in the definition of a representative scenario ξ^*. Section 2.3 gives a brief explanation regarding how to select the input values and other feature generation possibilities, whereas section 3.4 provides a more detailed explanation for the stochastic capacitated facility location problem.

2.2 Representative scenario

The authors in [2] propose a bilevel program B to find a candidate representative scenario $\xi^* = (w, U, v)$. The candidate representative scenario $\xi^* = (w, U, v)$ is in fact a representative scenario only if the objective value of program B equals the objective value of program P. The outer program has control over the second-stage decision $y(\xi)$ and components U and v of the representative scenario, whereas the inner program has control over the first-stage decision x and component w of the representative scenario. Intuitively, the outer program adjusts U and v to obtain a better first-stage decision x while making sure that the second-stage constraints hold for the realizations of the random variable ξ in the support set Ξ, whereas the inner program determines the best first-stage decision x and component w given the values of components U and v.

$$\begin{aligned}
B: \quad \min_{\substack{y(\xi)\in\mathbb{R}^m, v\in\mathbb{R}^q,\\ U\in\mathbb{R}^{q\times n}}} \quad & c^T x + \frac{1}{|\Xi|}\sum_{\xi\in\Xi} q(\xi)^T y(\xi) && \text{(5a)}\\
\text{subject to:} \quad & Wy(\xi) \le h(\xi) - T(\xi)x \quad \forall \xi \in \Xi && \text{(5b)}\\
& y(\xi)_i \in \mathbb{Z} \quad \forall \xi \in \Xi, \forall i \in \mathcal{I}_2 && \text{(5c)}\\
& (x,w) \in \operatorname*{argmin}_{x\in\mathbb{R}^n, w\in\mathbb{R}^m} \quad c^T x + w^T v && \text{(5d)}\\
& \text{subject to:} \quad Ax \le b && \text{(5e)}\\
& \qquad x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}_1 && \text{(5f)}\\
& \qquad Ww \le v - Ux && \text{(5g)}\\
& \qquad w_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}_2 && \text{(5h)}
\end{aligned}$$

Even though program B finds a representative scenario as long as one exists, it is a truly hard program to solve. Bilevel programs are known to be hard to solve when decision variables are continuous, and the addition of integrality constraints only increases the difficulty to solve them [5]. Thus, although desirable, solving program B to find representative scenarios, even only for the generation of training data, is currently not feasible in practice.

In this sense, the authors in [2] propose an heuristic to build representative scenarios for the generation of the training data for the linear regression model. The main idea of the heuristic for some instance $p = (A, b, c, W, \Xi)$ is to first solve program P and store the objective value o^P. Next, the heuristic creates an artificial realization $\bar{\xi}$ equal to the average realization of the random variable ξ, solves program S with the artificial realization $\bar{\xi}$, and stores the objective value $o^S(\bar{\xi})$. If $o^S(\bar{\xi})$ is sufficiently close to o^P, the heuristic returns the artificial realization $\bar{\xi}$ as a representative scenario ξ^*. If $o^S(\bar{\xi})$ is not sufficiently close to o^P, the heuristic perturbs $\bar{\xi}$ following some perturbation rule and check whether the new artificial realization $\bar{\xi}$ satisfies the conditions to be a representative scenario ξ^*. In this project, sufficiently close means $o^S(\bar{\xi})$ being within a 1% margin of o^P. This iterative procedure continues until a representative scenario has been found or until a stop criteria has been met, namely a maximum number of iterations. Note that the heuristic may not find a representative scenario for some instances after a maximum number of iterations.

2.3 Feature generation

It is important to think carefully about the input values given to a linear regression model. Some instance features may be more relevant than others, and ignoring useless features can improve the performance of the linear regression model by decreasing the dimension of the input. In addition, some features may have more relevance to the linear regression model after going through some type of processing rather than appearing in a raw form. For example, consider the coefficient matrix A. It may be better to provide as input to the linear regression model the maximum, minimum and average coefficients instead of the complete matrix depending on the interpretations of the application. Therefore, it follows that the decision regarding the input values of the linear regression model given some instance p depend a lot on the problem at hand.

2.4 Linear Regression

Once there is (i) a set of k pairs $(p_1, \xi_1^*), (p_2, \xi_2^*), ...(p_k, \xi_k^*)$ found through heuristics to train the linear regression model and (ii) a well established set of input values for the linear regression model according to the instance and the problem at hand, it becomes possible to train the linear regression model. The authors in [2] employ a linear regression model without additional modifications.

In this project, the Ridge regression has been employed to generate the linear regression model. The Ridge regression is similar to the (pure) linear regression, the only difference being a penalization term in the loss function to avoid having large coefficient values for certain features. The loss function minimizes the mean squared error. The penalization term has an attached weight factor, which can be set to larger or smaller values according to the desirable importance of the penalization term to the minimization. This project considers a small weight factor, namely 10^{-5}, to have a fair comparison with the results from [2] that use (pure) linear regression. For more information on how the Ridge regression works, see [7].

3 Facility location

This section presents the stochastic capacitated facility location problem, which has been chosen to study the performance of the algorithm, and explains additional aspects of the implementation of the RS algorithm for this problem.

3.1 Description

Capacitated facility location problems aim to select some locations from a set of candidate locations to receive capacitated facilities and determine the production capacity at each facility. The set of locations selected to receive capacitated facilities must produce enough units to meet the demand for a product in all locations. Locations with a facility have the demand satisfied by its own facility, whereas locations without a facility must receive units from facilities installed elsewhere. In addition, these problems often determine which facility will be attending the demand of which location according to transportation constraints.

Consider the following example with four locations presented in figure 1, where location 1 has a demand of 6 units, location 2 has a demand of 9 units, location 3 has a demand of 2 units, and location 4 has a demand of 1 units. If there are no additional constraints, one solution for this instance would be to install a facility at location 2 with capacity of 18 units, and attend the demand for units of locations 1, 3, and 4. This solution can be seen on figure 2.

The objective function often minimize the installation and transportation costs. Therefore, the optimal solution of capacitated facility location problems highly depend on the installation and transportation costs, as well as additional

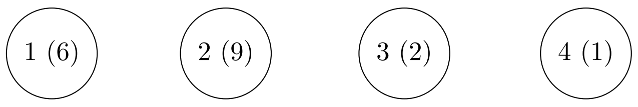

Fig. 1: Example of capacitated facility location instance.

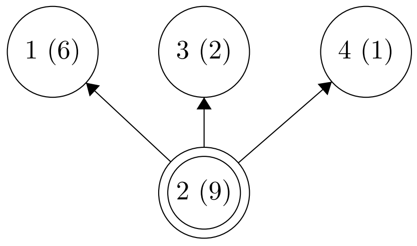

Fig. 2: Example of capacitated facility location solution.

constraints from the application. In the stochastic setting, the demand values of the locations are not known during the design of the facilities, and thus depend on the realization of a random variable.

Stochastic capacitated facility location problems describe a wide range of real-world applications, going from the once-in-a-lifetime installation of factories throughout the country to the daily installation of commercial booths throughout the city. For the sake of consistency, assume that the application of the stochastic capacitated facility location problem studied in this project needs to choose locations to install facilities frequently and quickly, which justifies the effort of applying the RS algorithm. Otherwise, it would arguably not be worth the effort to use the RS algorithm for only one first-stage decision.

3.2 Formulation

Let $\mathcal{L} = \{1, ...n\}$ be the set of n locations. The stochastic capacitated facility location problem has two sets of first-stage decision variables : $b_i \in \{0,1\} \, \forall i \in \mathcal{L}$, which equals 1 if there is a facility at location i and 0 otherwise, and $v_i \in \mathbb{R}^+$, $\forall i \in \mathcal{L}$, which stores the production capacity in units of a facility installed at location i. The stochastic capacitated facility location problem has two sets of second-stage decision variables: $u_{ij} \in \{0,1\} \, \forall i \in \mathcal{L}, \forall j \in \mathcal{L}$, which equals 1 if a facility at location i transports units to location j and 0 otherwise, and $y_{ij} \in \mathbb{R}^+ \, \forall i \in \mathcal{L}, \forall j \in \mathcal{L}$, which stores the number of units transported from location i to location j. There are four set of deterministic parameters: the fixed cost of installing a facility at location i given by $c_i^f \in \mathbb{R}^+ \, \forall i \in \mathcal{L}$, the production cost of one unit for a facility at location i given by $c_i^v \in \mathbb{R}^+ \, \forall i \in \mathcal{L}$, the fixed cost of transporting from location i to location j given by $c_{ij}^{tf} \in \mathbb{R}^+ \, \forall i \in \mathcal{L}, \forall j \in \mathcal{L}$, and the transportation cost of one unit from location i to location j given by $c_{ij}^{tv} \in \mathbb{R}^+ \, \forall i \in \mathcal{L}, \forall j \in \mathcal{L}$. Lastly, there is one set of stochastic parameters: $d_j(\xi) \in \mathbb{R}^+ \, \forall j \in \mathcal{L}, \forall \xi \in \Xi$, which is the demand value in units at location j according to the realization of the random variable ξ.

In this sense, it is possible to explicitly write programs P, S, and D for the stochastic capacitated facility location problem (SCFLP), i.e., programs $SCFLP^P$, $SCFLP^D$ and $SCFLP^S$ as follows. Since these programs describe the same application, only program $SCFLP^P$ will be explained in detail. The objective function (6a) minimizes installation and transportation costs taking into consideration the expected value of the random variable. Constraint (6b) requires that at least 10% and at most 75% of the locations must receive facilities. Constraints (6c) require that a facility must be open to have a production capacity greater than zero. Constraints (6e) guarantee that the number of units transported from a location i is less or equal than the production capacity. Constraints (6f) require that the demand at each location must be met, either by producing units in the same location or by receiving it from other locations. Lastly, constraints (6g) guarantee that there is a transportation between two locations only if this transportation channel has been opened.

$$
\begin{aligned}
SCFLP^P: \quad & \min_{b\in\{0,1\}^n, v\in\mathbb{R}^n} && \sum_{i\in\mathcal{L}} (c_i^f b_i + c_i^v v_i) + \mathbb{E}_\xi[Q(x,\xi)] && \text{(6a)}\\
& \text{subject to:} && 0.1n \le \sum_{i\in\mathcal{L}} b_i \le 0.75n && \text{(6b)}\\
& && v_i \le M b_i && \text{(6c)}\\
\text{where} \quad Q(x,\xi): \quad & \min_{u\in\{0,1\}^{n\times n}, y\in\mathbb{R}^{n\times n}} && \sum_{i\in\mathcal{L}}\sum_{j\in\mathcal{L}} (c_{ij}^{tf} u_{ij} + c_{ij}^{tv} y_{ij}) && \text{(6d)}\\
& \text{subject to:} && \sum_{j\in\mathcal{L}} y_{ij} \le v_i && \text{(6e)}\\
& && \sum_{i\in\mathcal{L}} y_{ij} = d_j(\xi) && \text{(6f)}\\
& && y_{ij} \le M u_{ij} && \text{(6g)}
\end{aligned}
$$

$$
\begin{aligned}
SCFLP^D: \quad & \min_{\substack{b\in\{0,1\}^n, v\in\mathbb{R}^n\\ u(\xi)\in\{0,1\}^{n\times n},\\ y(\xi)\in\mathbb{R}^{n\times n}}} && \sum_{i\in\mathcal{L}} (c_i^f b_i + c_i^v v_i) + \frac{1}{|\Xi|}\sum_{\xi\in\Xi}\sum_{i\in\mathcal{L}}\sum_{j\in\mathcal{L}} (c_{ij}^{tf} u_{ij} + c_{ij}^{tv} y_{ij}) && \text{(7a)}\\
& \text{subject to:} && 0.1n \le \sum_{i\in\mathcal{L}} b_i \le 0.75n && \text{(7b)}\\
& && v_i \le M b_i && \text{(7c)}\\
& && \sum_{j\in\mathcal{L}} y(\xi)_{ij} \le v_i && \text{(7d)}\\
& && \sum_{i\in\mathcal{L}} y(\xi)_{ij} = d_j(\xi) && \text{(7e)}\\
& && y(\xi)_{ij} \le M u(\xi)_{ij} && \text{(7f)}
\end{aligned}
$$

$$SCFLP^S: \quad \min_{\substack{b\in\{0,1\}^n, v\in\mathbb{R}^n,\\ u\in\{0,1\}^{n\times n}, y\in\mathbb{R}^{n\times n}}} \quad \sum_{i\in\mathcal{L}}(c_i^f b_i + c_i^v v_i) + \sum_{i\in\mathcal{L}}\sum_{j\in\mathcal{L}}(c_{ij}^{tf} u_{ij} + c_{ij}^{tv} y_{ij}) \tag{8a}$$

$$\text{subject to:} \quad 0.1n \leq \sum_{i\in\mathcal{L}} b_i \leq 0.75n \tag{8b}$$

$$v_i \leq M b_i \tag{8c}$$

$$\sum_{j\in\mathcal{L}} y_{ij} \leq v_i \tag{8d}$$

$$\sum_{i\in\mathcal{L}} y_{ij} = d_j^* \tag{8e}$$

$$y_{ij} \leq M u_{ij} \tag{8f}$$

3.3 Assumptions

It is important to guarantee that program $SCFLP^P$ has complete recourse and finite support. Note that the complete recourse assumption does not hold, for example, once the first-stage decision is set as $v_i = 0, b_i = 0\,\forall i \in \mathcal{L}$ because constraints (6f) cannot be satisfied with equality. In this sense, it is possible to guarantee the complete recourse property by artificially adding a hub facility with infinite production capacity and high transportation costs to the locations. Intuitively, this hub can be seen as another facility that is far away from the candidate locations but could meet the demand if no facilities have been installed. Due to the high transportation costs, the locations will never prefer to receive units from the hub facility because the transportation costs will always be more expensive than receiving from closer facilities. Therefore, the locations will only turn to the hub facility if there are no facilities installed close to the candidate locations. In this project, this change in program $SCFLP^P$ to have a hub facility has been used to guarantee the complete recourse assumption.

In turn, the finite support assumption may not hold according to the facility location application. In this sense, the only option to guarantee a finite support Ξ is to sample the original continuous support of the random variable ξ according to some sampling technique. In this project, the instances of the stochastic capacitated facility location program have been generated taking into consideration that the random variable has a finite support.

3.4 Instances

The instances of the stochastic capacitated facility location problem have been generated artificially following the procedure presented in [2]. The instances have $n = 10$ locations, and $|\Xi| = 50$ feasible realizations of the random variable. In short, all instances have the same values for c_{ij}^{tf} and c_{ij}^{tv} and different values for c_i^f, c_i^v, and $d_j(\xi)$. The values of c_{ij}^{tf}, c_{ij}^{tv}, c_i^f, c_i^v, and $d_j(\xi)$ have been defined as follows, where $\mathcal{U}(a, b)$ denotes a uniform distribution between a and b, and

$\mathcal{P}(c)$ denotes a Poisson distribution with expected value of c.

$$c_{ij}^{tf} = 10 \cdot abs(i-j) \quad \forall i \in \mathcal{L}, \forall j \in \mathcal{L} \tag{9a}$$

$$c_{ij}^{tv} = abs(i-j) \quad \forall i \in \mathcal{L}, \forall j \in \mathcal{L} \tag{9b}$$

$$c_i^f \sim \mathcal{U}(15, 20) \quad \forall i \in \mathcal{L} \tag{9c}$$

$$c_i^v \sim \mathcal{U}(5, 10) \quad \forall i \in \mathcal{L} \tag{9d}$$

$$\lambda_i = \frac{c_i^f + 10 \cdot c_i^v}{\sqrt{n}} \quad \forall i \in \mathcal{L} \tag{9e}$$

$$d_i(\xi) \sim \mathcal{P}(\lambda_j) \quad \forall j \in \mathcal{L} \tag{9f}$$

A total of 50000 instances were randomly generated. The instances have been labeled from 1 to 50000, where p_k denotes the k-th instance. The heuristic explained in section 2.2 managed to generate representative scenarios for 49636 instances. More details about the perturbation of the candidate representative scenario can be found in the attached notebook. Let the 49636 instances with representative scenarios be referred to as set Λ, and the remaining 364 instances without representative scenarios be referred to as set Γ. Set Λ will be later split into two sets, where one will be used to train the linear regression model and the other will be used test the performance of the LR approach. In turn, set Γ will be used as a sanity check to see how well the RS algorithm performs for instances where heuristics did not manage to find a representative scenario, i.e., instances that are likely harder to find a representative scenario.

Regarding the selection of input values for the linear regression model, the values of c_{ij}^{tf} and c_{ij}^{tv} $\forall i \in \mathcal{L}, \forall j \in \mathcal{L}$ are not given as input because they do not vary between instances by construction. The values of c_i^f and c_i^v $\forall i \in \mathcal{L}$ are given as input without additional modifications, so they represent 20 distinct input values. In turn, the values of $d_j(\xi)$ $\forall j \in \mathcal{L}, \forall \xi \in \Xi$ are given as input after a feature generation process. Note that Ξ can be seen as matrix with n rows, one for each location, and $|\Xi|$ columns, where $|\Xi|$ is the number of realizations of the random variable. Instead of providing Ξ as a raw input, i.e., $n \cdot |Xi|$ distinct values, the authors in [2] provide the maximum, the minimum, the average, the standard deviation, the median, the 75th quantile and the 25th quantile of each row of the matrix, i.e., of each location. There are 7 statistics per location, which gives 70 distinct input values. In addition, the authors provide as input values the percentage of the realizations where a certain fraction f of the demand at a certain location i is greater than and less than the demand on the other locations. Intuitively, this metric gives some information about either a dominant or submissive behavior of a location in comparison to other locations. There are 5 values of f, namely $f \in \{0.9, 1.0, 1.1, 1.2, 1.5\}$, 2 types of comparison (greater than or lesser than), and 10 locations, thus resulting in 100 distinct input values. In the end, there are $20 + 70 + 100 = 190$ inputs values for the linear regression model given a certain instance p.

4 Experiments

This section discusses the computational performance of the RS algorithm with the LR approach in comparison to (i) the optimal solution of the deterministic model given by Gurobi and (ii) other solutions given by other intuitive approaches to succinctly describe the support set Ξ.

4.1 Computational environment

The code has been written in Julia 1.5.3 with the help of a few packages. The `JuMP` package has been used to model the mixed-integer programs, whereas the `Gurobi` package has been used to solve the mixed-integer programs. The `MultivariateStats` package has been used to train the linear regression model following the Ridge regression. The experiments have been done in a personal computer with processor Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30GHz and operational system Windows 10. More details about the implementation of the RS algorithm can be found in the attached notebook.

4.2 Preliminary computations

The computational experiments rely on some preliminary computations, namely:

- The generation of 50000 instances according to section 3.4;
- The solution of program $SCFLP^D$ for all 50000 instances to assist the search for representative scenarios. Program $SCFLP^D$ has been solved up to a proven optimality gap of 2% by the Gurobi solver;
- The generation of representative scenarios for as much instances as possible through heuristics according to section 2.2. The heuristics found representative scenarios for 49636 instances out of 50000 instances.

Since these preliminary computations took a long time for 50000 instances, they were done in different moments and the outcomes, namely the instance files, the Gurobi outputs, and the representative scenarios, have been written on local files. More details about storage and use of these files for the generation of the computational results can be found in the attached notebook.

4.3 Linear regression model

Instance set Λ, i.e., the 49636 instances with a representative scenario found by the heuristic, has been randomly split into two sets: the training set with 45000 instances that will be used to train the linear regression model, and the testing set with the remaining 4636 instances that will be used to test the performance of the LR approach in finding near-optimal first-stage decisions.

The training of the linear regression model using the Ridge regression has been conducted offline, i.e., in a step prior to the actual utilization of the RS algorithm. As highlighted in section 3.1, this project considers a facility location problem where the selection of locations to receive capacitated facilities happens frequently, thus justifying the efforts of generating data and training a linear regression model to find faster near-optimal solutions through the RS algorithm. More details about the training of the linear regression model with Ridge regression can be found in the attached notebook. For the sake of reproducibility, this project has generated the linear regression model once and stored it. The offline training of the linear regression model has taken 5 minutes.

4.4 Computational approaches

This project considers the following computational approaches:

GRB: The Gurobi (GRB) approach solves program D with the complete instance information to obtain a respective first-stage decision x. Therefore, the GRB approach finds the optimal first-stage decision x that the other approaches would like to come close to by solving a smaller program.

LR: The linear regression (LR) approach guesses a representative scenario ξ^* based on the instance information and solves program S with it as the representative scenario ξ^* to obtain a respective first-stage decision x. The LR approach summarizes the performance of the RS algorithm, which should be able to produce a near-optimal first-stage decision x [2].

RND: The random (RND) approach draws a random realization of the random variable ξ from the support set Ξ and solves program S with it as the representative scenario ξ^* to obtain a respective first-stage decision x. The RND approach has been proposed as one of the sanity checks in [2].

AVG: The average (AVG) approach takes the average realization of the random variable ξ from the support set Ξ and solves program S with it as the representative scenario ξ^* to obtain a respective first-stage decision x. The AVG approach has been proposed as one of the sanity checks in [2].

HYB: The hybrid (HYB) approach builds an artificial support set $\bar{\Xi}$ with three realizations (one guessed by the linear regression model, one random realization, and one average realization) and solves program D with it as the support set Ξ to obtain a respective first-stage decision x. The HYB approach has been proposed in this project after an inspiration from ensemble learning, where multiple prediction models are used to take a decision [8]. In this sense, the random and average realizations can be seen as prediction models in addition to the linear regression model.

APR(k): The approximate approach (APR(k)) builds an artificial support set $\bar{\Xi}$ with k random realizations of the random variable ξ and solves program D with it as the support set Ξ to obtain a respective first-stage decision x. The APR(k) approach has been proposed in this project after an inspiration from the sampling techniques, where the information of the support set Ξ is approximated by a random subset of realizations.

4.5 Computational comparison

The comparison between computational approaches uses two main indicators. The first indicator is the computational time to find the respective first-stage decision $\bar{x}$. In practice, this is the time it would take to find the first-stage decision $\bar{x}$ that will be used by decision-makers to install capacitated facilities. The second indicator is how far the objective value induced by the first-stage decision $\bar{x}$ outputted by the approach is from the optimal solution, i.e., how far is the optimal solution from program D with first-stage decisions fixed to $\bar{x}$ from the optimal solution from program D. Intuitively, this gives us an idea of how good is the first-stage decision $\bar{x}$ outputted by the approach.

The time indicator has been calculated using the `@elapsed` macro, which returns the time it takes to execute a certain function in seconds. Note that Julia compiles functions as they are called. Thus, the notebook executes all the approaches at least once before they are actually evaluated to avoid having compilation time as noise. The evaluation of time considers all the steps involved in running a certain approach, i.e., the time it takes to load the instance, gather necessary information, build the program, find the optimal solution and return the first-stage decision. The major differences between approaches are (i) the time it takes to find an optimal solution, which will highly depend on the size of the program, and (ii) the time it takes to gather necessary information for the approach. For example, the LR approach needs to retrieve from files the previously trained linear regression model, convert the instance into an input of the linear regression model, and guess the representative scenario. Likewise, the other approaches conduct necessary steps to complete their computations.

The ratio indicator has been calculated using the following formula, where $\mathit{objective}(\mathsf{approach})$ denotes the objective value of program D with the fixed first-stage decision $\bar{x}$ given by the approach. In turn, *optimal* denotes the objective value of program D solved up to a proven optimality gap of 2%. If a certain `approach` has $\mathit{ratio}(\mathsf{approach})$ close to zero, it is possible to affirm that the fixed-stage decision $\bar{x}$ given by the `approach` is near-optimal. In turn, if a certain `approach` has $\mathit{ratio}(\mathsf{approach})$ far from zero, it is possible to affirm that the fixed-stage decision $\bar{x}$ given by the approach `approach` has bad quality, as it forces solutions that are far from the optimal solution. Note that since *optimal* stores the optimal value of program D solved up to a proven optimality gap of 2% for an instance, it is possible that $\mathit{ratio}(\mathsf{approach})$ is negative for a certain approach. This means that the `approach` manages to find a first-stage decision $\bar{x}$ even better than the sufficiently optimal, which is an quite interesting fact.

$$\mathit{ratio}(\mathsf{approach}) = \frac{\mathit{objective}(\mathsf{approach}) - \mathit{optimal}}{\mathit{optimal}} \tag{10}$$

In addition, it is possible to talk about the stability of an approach in terms of the objective ratio or computational time. The main idea is that a stable approach has small standard deviation and well behaved outlier cases, i.e., the maximum, the minimum and the median of a distribution are as close as possible. In practice, stable approaches are nicer because they are unlikely to output a very bad or a very slow first-stage decision from time to time, and thus provide better information to the decision makers.

4.6 Experimental questions

There are four main experimental questions that should be answered in this project through the computational experiments performed in the next section:

Question 1 Are the computational results of this project reasonably close to the computational results of [2]? Even though there are parts that cannot be reproduced exactly or are not clearly stated, close computational results in comparison to [2] show that the implementation tends to be correct. Here, the testing set from the instance set Λ will be used.

Question 2 Can the LR approach provide faster near-optimal first-stage decisions? If the LR approach manages to provide faster near-optimal first-stage decisions, it would make sense to choose to work with the RS algorithm. Here, the testing set from the instance set Λ will be used.

Question 3 How does the LR approach compare to the APR(k) approach? The use of linear regression models inevitably brings additional complexity to the decision making process without providing strong theoretical results. Therefore, it makes sense to prefer the LR approach only if it provides faster, more stable, and better solutions than the APR(k) approach. Otherwise, the APR(k) approach has better theoretical guarantees, as well as clear directions for improvements, and should be preferred over the LR approach. Here, the testing set from the instance set Λ will be used.

Question 4 How does the RS algorithm perform for the instance set Γ, i.e., the instances where the heuristic did not manage to find representative scenarios? These instances have been disconsidered in the computational experiments of [2], but they might show realistically what would happen with unknown (as they were not part of the training data) and hard (as the heuristic did not find representative scenarios for them) instances.

4.7 Experimental results

Figure 3 shows the relationship between the **average** objective ratio and the **average** computational time per approach. Table 1 presents statistics on the objective ratio per approach, whereas table 2 presents statistics on the computational time per approach. Figure 4 shows the same information as figure 3 but only for the GRB, LR, RND, and AVG approaches that appear in the reference results [2]. Tables 3 and 4 also present the same information as tables 1 and 2, respectively, but only for the GRB, LR, RND, and AVG approaches that appear in the reference results [2]. The results from [2] have been reproduced and slightly simplified here to simplify the comparison.

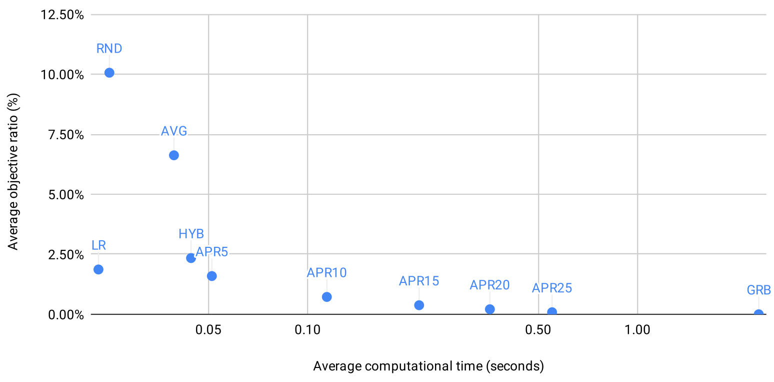

Fig. 3: Average objective ratio versus average computational time per approach.

Question 1. Let us analyze whether the computational results of this project are reasonably close to the computational results of [2] for the GRB, LR, RND, and AVG approaches.

Ratio (%)	GRB	AVG	RND	LR	HYB
Average	0.00%	6.62%	10.06%	1.87%	2.34%
Standard deviation	0.00%	1.42%	10.24%	0.62%	0.83%
Maximum	0.00%	12.26%	70.34%	4.57%	6.48%
Minimum	0.00%	2.11%	-0.42%	-0.04%	-0.12%
Median	0.00%	6.59%	6.61%	1.84%	2.29%
Ratio (%)	**APR5**	**APR10**	**APR15**	**APR20**	**APR25**
Average	1.59%	0.72%	0.38%	0.21%	0.08%
Standard deviation	1.79%	0.99%	0.68%	0.53%	0.40%
Maximum	21.27%	10.72%	8.02%	5.12%	3.19%
Minimum	-1.00%	-1.03%	-0.78%	-0.83%	-0.83%
Median	1.09%	0.46%	0.20%	0.08%	0.00%

Tab. 1: Statistics on the objective ratio per approach.

Time (seconds)	GRB	AVG	RND	LR	HYB
Average	2.33154	0.03929	0.02503	0.02319	0.04427
Standard deviation	2.91416	0.02962	0.01644	0.01251	0.01980
Maximum	40.07399	0.45742	0.10364	0.09614	0.15008
Minimum	0.21771	0.01231	0.00654	0.00888	0.01239
Median	1.28877	0.03393	0.01588	0.01992	0.04099
Time (seconds)	**APR5**	**APR10**	**APR15**	**APR20**	**APR25**
Average	0.05120	0.11428	0.21767	0.35674	0.55075
Standard deviation	0.02164	0.05774	0.11233	0.20949	0.42193
Maximum	0.17983	0.54470	1.33396	2.60954	5.79923
Minimum	0.01544	0.02993	0.04378	0.06171	0.11843
Median	0.04371	0.09349	0.21012	0.32425	0.45012

Tab. 2: Statistics the computational time per approach.

Ratio (%)	GRB	AVG	RND	LR
Average	0.00%	8.10%	12.10%	0.64%
Standard deviation	0.00%	1.59%	12.11%	0.41%
Maximum	0.00%	14.23%	94.42%	2.64%
Minimum	0.00%	3.36%	-0.33%	-0.62%
Median	0.00%	8.08%	8.24%	0.60%

Tab. 3: Statistics on the objective ratio per approach extracted from [2].

Figure 3 shows that there is a similar relationship between the GRB, LR, RND, and AVG approaches in comparison to the reference results in figure 4, i.e., the GRB approach solves the instance to optimality while taking a considerably larger time, the AVG and RND approaches solve the instance quickly but output a first-stage decision that is far from optimal, and the LR approach finds a near-optimal solution considerably fast on average.

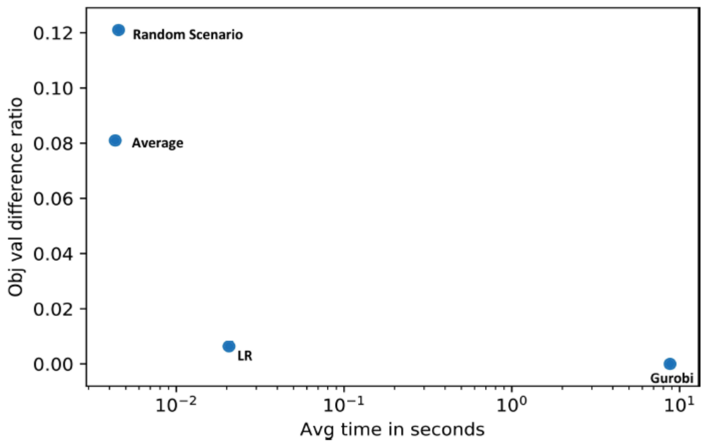

Fig. 4: Average objective ratio versus average computational time per approach extracted from [2].

Time (seconds)	GRB	AVG	RND	LR
Average	8.7713	0.0043	0.0045	0.0206
Standard deviation	17.919	0.0001	0.0003	0.0019
Maximum	559.32	0.0077	0.0081	0.0454
Minimum	0.4354	0.0041	0.0041	0.0194
Median	2.2621	0.0043	0.0044	0.0200

Tab. 4: Statistics the computational time per approach extracted from [2].

However, the GRB approach takes considerably less time to solve instances within this project, whereas the RND and AVG approaches take considerably more time to solve the instances. The first explanation for this observation could be that the instances used within this project were built following the same procedure, but they were not exactly the same. Slightly easier or more difficult instances could explain the difference in computational times for the GRB approach, as the structure of the instance could inherently allow the use of additional cuts or heuristics, helping the solver to achieve the optimal solution faster within this project. The second explanation for this observation could be that the authors in [2] did not count the process to compute the average scenario, as well as the process to sample the random scenario, as part of the computational time. The authors in [2] do not explicitly stress this information about the RND and AVG approaches in the paper, they only explain that for the LR approach they consider the time it takes to compute the representative scenario with the linear regression model. Therefore, it is difficult to know for sure what has been counted in the computational time. Note that for the LR approach, where there is a through explanation of how the computational time has been computed, the computational time within this project matches the computational time of [2].

In terms of average objective ratio seen on figures 3 and 4, there is no strong differences between what has been found in the reference paper and what has been found in this project. This project finds a objective ratio that is approximately 2 percentage points lower for the RND approach, 2 percentage points lower for the AVG approach, and 1 percentage point higher for the LR approach. Again, these small changes in the objective ratio can be explained by the fact that within this project the instances are not exactly equal to the instances they have used in the reference paper.

Tables 1 and 3 have similar results overall with a difference of at most 2 percentage points in almost all statistics. The only statistic with a high difference is the maximum objective ratio of the RND approach, where this project finds 70.34% and the reference paper finds 94.42%. Note that this statistic highly depends on the randomly sampled scenario, thus not giving enough reasons to question the validity of the results in this project. Overall, the RND and AVG approaches have better performance within this project, i.e., smaller average, standard deviation, maximum, minimum, and median of the objective ratio. In turn, the LR approach performs slightly worse within this project, i.e., a 1 percentage point higher average, a 0.2 percentage points higher standard deviation, a 2 percentage points higher maximum, a 0.6 percentage point higher minimum, and a 1.2 percentage points higher median objective ratio.

The difficulty to count the computational time per approach using the same pattern as the reference paper explains the difference between tables 2 and 4. Even though the LR approach within this project has computational time statistics similar to the reference results [2], the GRB approach takes considerably less time within this project, whereas the RND and AVG approaches take considerably more time within this project. This difference appears not only in the average computational time but also in the other statistics. In addition to the previously presented explanations, note that the authors in [2] do not define the computational language they have used for the implementation, which could also play an important role in determining the computational time.

In the end, even though there are some differences, it is possible to affirm that the computational results of this project are reasonably close to the computational results of [2] for the GRB, LR, RND, and AVG approaches.

Question 2. Let us analyze whether the LR approach can provide faster near-optimal first-stage decisions.

The authors in [2] come to the conclusion that the LR approach can provide faster near-optimal first-stage decisions by noticing that the objective ratio is considerably close to zero and the computational time is much faster than other approaches. In this sense, it is possible to affirm that the LR approach also provides faster near-optimal first-stage decisions within the scope of this project. The LR approach has an average objective ratio of 1.87%, which is not as small as the reference result in table 4 but still small, with a small standard deviation of 0.62%. At most, the LR approach provides a objective ratio of 4.57%, which is not as large if compared to the other approaches (only the APR(25) approach beats the LR approach in this sense). In terms of statistics, the LR approach seems to have a stable performance overall. In terms of computational time, the LR approach performs much better than the GRB approach even though the GRB approach is considerably faster within the scope of this project.

The RND and AVG approaches can be solved as quickly as the LR approach as seen on table 2, but their performance is considerably unstable as seen on table 1. In addition to the RND and AVG approaches, this project considers the HYB approach. Ideally, the HYB approach would be able to mix the information given by the LR, RND, and AVG approaches and provide an even better estimation of the support set Ξ. It is possible to see that the HYB approach performs better than the RND and AVG approaches, i.e., (i) the average objective ratio of the HYB approach is smaller than the average objective ratio of the AVG and RND approaches, (ii) the HYB approach seems to be more stable than the AVG and RND approaches, and (iii) the HYB approach is only slightly slower than the AVG and RND approaches. However, the HYB approach performs worse than the LR approach in all criteria. The average, standard deviation, median, maximum, and minimum of the HYB approach are higher than the LP approach, and the HYB approach also takes more computational time to output a first-stage decision. Therefore, the HYB approach cannot substitute the LR approach as the best option. Note that in ensemble learning the quality of the prediction models involved in the decision-making process plays an important role in the quality of the final solution. Therefore, it may still be possible to obtain better results with an ensemble learning based approach by using better prediction models than the average or random realizations.

In the end, it is possible to affirm that the LR approach can provide near-optimal first-stage decisions according to the results in this project.

Question 3. Let us analyze how the LR approach compares to the APR(k) approach.

The APR(k) approach follows the idea that the support set of the random variable can be approximated by sampling a percentage of the realizations. There is an inherent trade-off between (i) sampling a large number of scenarios and approximating well the support set and (ii) generating hard mixed-integer models that cannot be solved quickly. Therefore, the percentage of scenarios sampled from the support set plays an important role in determining whether the approximation is good and whether the mixed-integer program can be solved quickly. In this project, since there are 50 realizations of the random variable, the approximate approaches sample 10% (APR(5)), 20% (APR(10)), 30% (APR(15)), 40% (APR(20)), and 50% (APR(25)) of the support set.

Table 1 shows that the average objective ratio of the APR(k) approaches decreases as the percentage of the support set increases as expected. In addition, the growth of the percentage of the support set increases the stability of the approach, as the standard deviation and maximum objective ratio decrease continuously. In turn, table 2 shows that the average computational time of the APR(k) approaches increases as the percentage of the support set increases. The computational time also becomes less stable with the growth of the percentage of the support set. Nevertheless, the approximate approaches seem to be competitive approaches to approximate the support set according to some quality standard, based on which the right value of k can be chosen.

The interesting question is whether it is better to use the LR approach or the APR(k) approaches in practice. In terms of objective ratio, the LR approach has an average objective ratio similar to the APR(5) approach, which is also the approximate approach with the most competitive computational time. However, the APR(5) approach seems to be less stable than the LR approach. First, there is a 1 percentage point difference in the standard deviation, and an approximately 1 percentage point difference in the median. In addition, the maximum of the APR(5) approach is considerably larger than the maximum of the LR approach. The other approximate approaches provide better and more stable results in terms of average and standard deviation, but the maximum remains being a problem. The fact that approximate approaches sample random scenarios make it difficult to not sample scenarios that do not describe well the support set and thus induce these bad maximum statistics for some cases.

In the end, the decision to use the LR approach or the APR(k) approaches becomes a decision that depends on the quality requirements of the application. If there is enough time and will to train a good linear regression model, the LR approach seems to provide a better result than the approximate approaches. However, there are no theoretical guarantees that the LR approach will find a good representative scenario and induce a near-optimal solution. On the other hand, the approximate approaches provide good results on average, but it may provide bad results from time to time. However, there are theoretical guarantees that continuously sampling the support set will guide the approximate approaches towards the optimal solution.

Question 4. Let us analyze how the RS algorithm performs for the instance set Γ, i.e., the instances where the heuristic did not manage to find representative scenarios.

Tables 5 and 6 present the same information as tables 1 and 2, respectively, but taking into consideration instance set Γ, i.e., 364 instances where the heuristic did not manage to find a representative scenario. The authors in [2] decided to not include these instances in the computational results without providing additional explanation. Nevertheless, the RS algorithm through the LR approach has a very similar performance for instance set Γ. Since the results in tables 5 and 6 are pretty similar to the results in tables 1 and 2, the discussion from the previous sections can be extended to instance set Γ. In the end, the RS algorithm seems to perform well for instance set Γ as well.

Ratio (%)	GRB	AVG	RND	LR	HYB
Average	0.00%	6.59%	10.89%	1.88%	2.34%
Standard deviation	0.00%	1.48%	10.36%	0.68%	0.88%
Maximum	0.00%	10.96%	53.58%	4.45%	6.95%
Minimum	0.00%	3.15%	-0.26%	-0.20%	0.58%
Median	0.00%	6.56%	7.25%	1.85%	2.28%
Ratio (%)	APR5	APR10	APR15	APR20	APR25
Average	1.63%	0.74%	0.36%	0.19%	0.06%
Standard deviation	1.66%	1.07%	0.65%	0.52%	0.44%
Maximum	11.89%	7.19%	3.06%	2.50%	2.44%
Minimum	-0.54%	-1.02%	-0.94%	-0.92%	-0.92%
Median	1.14%	0.45%	0.19%	0.06%	0.00%

Tab. 5: Statistics on the objective ratio per approach for instance set Γ.

5 Conclusion

This project has explained the representative scenario (RS) algorithm [2], implemented it in a Jupyter notebook attached to this project, validated its ability to successfully find near-optimal first-stage decisions to program P, checked its computational results presented by the authors in [2], and compared it with

Time (seconds)	GRB	AVG	RND	LR	HYB
Average	3.14477	0.03482	0.02846	0.02012	0.04655
Standard deviation	4.34163	0.01717	0.01803	0.01056	0.02447
Maximum	34.95183	0.16312	0.09225	0.07174	0.27002
Minimum	0.26119	0.01562	0.00787	0.01054	0.01513
Median	1.49915	0.02777	0.01906	0.01522	0.04034
Time (seconds)	**APR5**	**APR10**	**APR15**	**APR20**	**APR25**
Average	0.05698	0.13854	0.25837	0.42973	0.62634
Standard deviation	0.02664	0.07015	0.13131	0.26196	0.46906
Maximum	0.16156	0.46130	0.92612	2.36913	3.48619
Minimum	0.02060	0.04037	0.06090	0.08485	0.14953
Median	0.04652	0.11558	0.24642	0.37324	0.49106

Tab. 6: Statistics on the computational time per approach for instance set Γ.

some other intuitive approaches to succinctly describe the support set Ξ.

The LR approach has proved to generate near-optimal first-stage decisions within the scope of this project. In addition to having a smaller objective ratio, the LR approach has a stable performance overall when compared to the other approaches. The main drawback of the LR approach is the difficulty to produce and maintain prediction models for certain problems, as well as guarantee that the prediction models are general enough and always produce good quality solutions. There are also no strong theoretical guarantees.

The HYB and APR(k) approaches proposed within this project have provided interesting results, but they still loose to the LR approach either in terms of having a worse average objective ratio, having an unstable method, or having a worse computational time. Some directions of improvement for the HYB approach would be to use better prediction models to compose the HYB approach, i.e., use realizations predicted with other prediction models instead of using a random and an average realization. Some directions of improvement for the APR(k) approaches would be the smart sampling of realizations from the support set through machine learning techniques or Monte Carlo techniques. In this sense, the APR(k) approaches could achieve stabler performances and beat the LR approach in terms of stability, thus becoming a better approach overall.

Additional future work directions mentioned in [2] include applying the RS algorithm with the LR approach to other computational problems and check whether it performs as well. In addition, it is also important to study the conjecture of the paper, which is the foundation behind the RS algorithm, to find out whether there are specific conditions that guarantee the existence of a representative scenario. In this sense, the lack of theoretical ground behind the existence representative scenario would be solved at least for some cases and the RS algorithm would have stronger theoretical grounds.

References

[1] Shabbir Ahmed. Two-stage stochastic integer programming: A brief introduction. *Wiley encyclopedia of operations research and management science*,

2010.

[2] Yoshua Bengio, Emma Frejinger, Andrea Lodi, Rahul Patel, and Sriram Sankaranarayanan. A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 99–111. Springer, 2020.

[3] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[4] Kibaek Kim and Sanjay Mehrotra. A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. *Operations Research*, 63(6):1431–1451, 2015.

[5] Thomas Kleinert, Martine Labbé, Ivana Ljubić, and Martin Schmidt. A survey on mixed-integer programming techniques in bilevel optimization. 2021.

[6] Gilbert Laporte and François V Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142, 1993.

[7] Gary C McDonald. Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):93–100, 2009.

[8] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.