Algebra for Machine Learning and Stochastic Programming IV

Yuchen Ge

September 2022

Contents

Abstract

This article introduces some combinatorial methods combined with Gröbner Basis and Graver Basis for Stochastic Integer Programming.

1 Introduction

Traditionally machine learning and optimization are two different branches in computer science. They need to accomplish two different types of tasks, and they are studied by two different sets of domain experts. Machine learning is the task of extracting a model from the data, while optimization is to find the optimal solutions from the learned model. In the current era of big data and AI, however, such separation may hurt the end-to-end performance from data to optimization in unexpected ways. The paradigm of data-driven optimization is to tightly integrate data sampling, machine learning, and optimization tasks. There are generally two approaches in this paradigm, one is optimization from structured samples, which carefully utilizes the structural information from the sample data to adjust the learning and optimization algorithms; the other is combinatorial online learning, which adds feedback loop from the optimization result to data sampling and learning to improve the sample efficiency and optimization efficacy. Stochastic Integer Programming is a typical example of data-driven optimization. I will describe a new efficient algorithm for purely combinatorial Stochastic Integer Programming.

1.1 Stochastic Integer Programming

Benders decomposition has enabled considerable progress in stochastic programming with continuous recourse [6] by iterating between a "master problem" and continuous "sub-problems". However, this approach is not designed to accommodate stochastic integer programs, which generally have non-convex and non-continuous recourse functions [7]. [8] and [9] provide comprehensive surveys on stochastic integer programming. Several decomposition approaches have been proposed, based on enumeration ([2]), branch and bound ([10]), branch and cut ([11]), Gomory cuts ([12]), split cuts ([13]), sample average approximation ([14]), disjunctive programming ([15]) and dynamic programming ([16]). The ideas of Benders decomposition have been extended to integer recourse functions using Lagrangian relaxation and branch-and-bound algorithms, but the resulting master problem is non-convex ([3]). One of the most widely applied solution approaches is the integer L-shaped method from [17]. This method generates optimality and feasibility cuts iteratively to the master problem by leveraging the optimal value of the second-stage objective function for any first-stage integer solution. This method was extended with non-linear and Gomory cuts ([18]), with specialized branching ([10]), with thin-direction branching and multi-cut heuristics ([24]), and with alternating cuts and a cut-generating optimization model ([20]).

These developments have enabled applications of stochastic integer programming to vehicle routing ([21]), capacity expansion ([22]), inventory management ([23]), nurse staffing and scheduling ([24]), etc. At the same time, the integer L-shaped method is most effective when the first-stage problem has a limited solution space. Many problems, however, admits an exponentially large first-stage solution space—so that the integer L-shaped algorithm did not converge even in small instances.

These challenges motivate new solution algorithms for stochastic integer programming. [25] proposes a novel decomposition approach that leverages the dual linear programming relaxation of the second-stage integer problem and its reduced costs. As such, their approach shares similarities with the additive bounding procedure from [26], which uses dual information from a model relaxation to derive a lower bound for NP-hard combinatorial optimization problems—with applications to the traveling salesman and vehicle routing problems ([27],[28], [29]).

1.2 Scenario Reduction

A challenge in stochastic programming involves defining uncertainty scenarios and their probabilities. Most applications rely on distributional assumptions regarding the uncertain parameters to sample scenarios. Scenario reduction methods have been proposed when the number of scenarios is too large to ensure computational tractability ([30], [31]). These methods start with a "true" probability distribution, and generate another probability distribution of prescribed cardinality that approximates it best. The scenario reduction problem is typically solved using heuristic algorithms. It has been applied, for instance, to the unit commitment and capacity expansion problems in energy planning ([32], [33]).

1.3 Problem Statement

Instead of SIP (A universally acknowledged definition is unavailable), we give the definition of Stochastic Mixed Integer Program. A Stochastic Mixed Integer Program (SMIP) is to solve

$$\begin{aligned} &\min c^T x + \mathbb{E}[Q(x,\xi)] \\ &\text{s.t. } Ax \geq b \\ &\quad x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \end{aligned} \tag{1}$$

where $\xi = (q, h, T, W)$ and

$$\begin{aligned} Q(x,\xi) = \min & q^T y \\ \text{s.t. } & Wy = h - Tx \\ & y \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2} \end{aligned}$$

where x denotes the first-stage decision variables and y denotes the second-stage decision variables and sometimes we assume n_1, p_1, n_2 , or p_2 are zero, which depends.

We assume the random data ξ is represented by a finite set of scenarios: $\left(q^i, h^i, T^i, W^i\right), i = 1, \ldots, N$, where scenario i occurs with probability p_i. So we have the following extensive form (deterministic equivalent) of an SMIP.

Definition 1.1. Deterministic equivalent of an SMIP is

$$\begin{aligned} \min \quad & c^T x + \textstyle\sum_{i=1}^N p_i q_i^T y_i \\ \text{s.t.} \quad & Ax \geq b \\ & T_i x + W_i y_i = h_i \\ & x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \\ & y_i \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \quad i = 1, \ldots, N \end{aligned} \tag{2}$$

When $n_1 = 0$ and $n_2 = 0$ (i.e. purely integral/combinatorial), we have a method [1] to deal with it. And we have another method [2] to deal with a Stochastic Program with Complete Integer Recourse, which is a SMIP when $p_1 = n_2 = 0$ (i.e. first-stage continuous and second-stage combinatorial). The above two existing methods require the matrix T_i, W_i to be invariant. However, in the proposal I will introduce a new algorithm for the case only when $n_1 = 0$ and $n_2 = 0$.

1.4 Stochastic Facility Location Problem

The stochastic model has many applications. For example, we present the Stochastic facility location: a firm is deciding which facilities to open to serve customers with random demands. Goal is to minimize total (expected) cost.

Notation:
- I: Set of possible facilities to open
- J: Set of customers
- f_i: Fixed cost for opening facility i
- C_i: Capacity of facility i
- c_{ij} : Unit cost for serving customer j demand at facility i
- q_j : Penalty per unit of unmet demand of customer j
- p_s : Probability of scenario s, $s = 1, \ldots, S$
- d_j^s: Demand of customer demand j in scenario s

Assume first-stage integer variables: $x_i = 1$ if facility i is open, 0 otherwise. Then we have the following SMIP:

$$\begin{aligned} &\min_x \textstyle\sum_{i \in I} f_i x_i + \sum_{s \in S} p_s Q_s(x) \\ &\text{s.t. } x_i \in \{0,1\}, \quad i \in I \end{aligned} \tag{3}$$

where

$$\begin{aligned} Q_s(x) = \min_{x,y} & \sum_{i\in I}\sum_{j\in J} c_{ij}y_{ij} + \sum_{j\in J} q_j z_j \\ \text{s.t. } & \sum_{i\in I} y_{ij} + z_j \geq d_j^s, \quad j \in J \\ & \sum_{j\in J} y_{ij} \leq C_i x_i, \quad i \in I \\ & y_{ij} \geq 0, z_j \geq 0, \quad i \in I, j \in J \end{aligned} \tag{4}$$

2 Preliminary on Integer Programming

2.1 Branch and Bound

First we fix some terminologies. Sub-problems (nodes) form a **search tree**. Eliminating a problem from further consideration is called **pruning**. The act of bounding and then branching is called **processing**. A sub-problem that has not yet been processed is called a **candidate**. The set of candidates is the **candidate list**. Then we have:

Algorithm 2.1. (Branch and Bound Algorithm)
Step 1: Derive an bound U using a heuristic method (if possible).
Step 2: Put the original problem on the candidate list.
Step 3: Select a problem S from the candidate list and solve the relaxation to obtain the bound $\ell(S)$
- Relaxation infeasible $\Rightarrow$ node can be pruned.
- $\ell(S) < U$ and the solution is feasible for the MIP $\Rightarrow$ set $U \leftarrow \ell(S)$.
- $\ell(S) \geq U \Rightarrow$ node can be pruned.
- Otherwise, branch. Add the new sub-problems to the list.

Step 4: If the candidate list in nonempty, go to Step 3. Otherwise, the algorithm is completed.

By adding Redundant constraints as below,

$$y_{ij} \leq \min\left\{d_j^s, C_i\right\} x_i, \quad \forall i \in I, j \in J$$

$$\begin{aligned} & \min_x \textstyle\sum_{i\in I} f_i x_i + \sum_{s\in S} p_s Q_s(x) \\ & \text{s.t. } x_i \in \{0,1\}, \quad i \in I \end{aligned} \tag{5}$$

where

$$\begin{aligned} Q_s(x) = \min_{x,y} & \sum_{i\in I}\sum_{j\in J} c_{ij}y_{ij} + \sum_{j\in J} q_j z_j \\ \text{s.t. } & \sum_{i\in I} y_{ij} + z_j \geq d_j^s, \quad j \in J \\ & \sum_{j\in J} y_{ij} \leq C_i x_i, \quad i \in I \\ & y_{ij} \leq \min\left\{d_j^s, C_i\right\} x_i, \quad \forall i \in I, j \in J \\ & y_{ij} \geq 0, z_j \geq 0, \quad i \in I, j \in J \end{aligned} \tag{6}$$

We have that for x_i, the set of integer feasible points satisfying these are the same. But many fractional points that satisfy original formulation do not satisfy the redundant constraints.

Here are some comments. Using the formulation with better bound is almost always (much) better, since we can prune more often. But one may need to use specialized techniques to solve relaxation problems with more constraints.

2.2 Valid Inequalities

Let $X = \left\{x \in \mathbb{R}_+^n : Ax \leq b, x_j \in \mathbb{Z}, j \in J\right\}$

Definition 2.2. An inequality $\pi x \leq \pi_0$ is a valid inequality for X if $\pi x \leq \pi_0$ for all $x \in X$. $(\pi \in \mathbb{R}^n, \pi_0 \in \mathbb{R})$

Here, valid inequalities are also called "cutting planes" or "cuts". The goal of adding valid inequalities to a formulation is to improve relaxation bound and explore fewer branch-and-bound nodes.

How to use them in a branch-and-bound algorithm? We can either add them to the initial formulation, or add them only as needed to cut off fractional solutions. When add them only as needed to cut off fractional solutions, we have two different situations.

(1) Cut-and-branch: Do this only with the initial LP relaxation (root node).

(2) Branch-and-cut: Do this at all nodes in the branch-and-bound tree.

Then we mainly focus on Branch-and-cut method: at each node in branch-and-bound tree we add valid inequalities we need.

Algorithm 2.3. (Branch-and-cut)
Step 1: Solve current LP relaxation $\Rightarrow \hat{x}$
Step 2: Attempt to generate valid inequalities that cut off $\hat{x}$
Step 3: If cuts found, add to LP relaxation and go to step 1

This approach is the **heart of all modern MIP solvers** because it reduces the number of nodes to explore with improved relaxation bounds and add inequalities required to define feasible region.

3 Branch and Cut Based Methods

Consider the SMIP where we introduce the variable

$$\begin{aligned} \min \quad & c^T x + \textstyle\sum_{s=1}^{S} p_s \theta_s \\ \text{s.t.} \quad & Ax \geq b \\ & \theta_s \geq Q_s(x), \quad s = 1, \ldots, S \\ & x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \end{aligned} \tag{7}$$

where for $s = 1, \ldots, S$

$$\begin{aligned} Q_s(x) = \min & q_s^T y \\ \text{s.t. } & W_s y = h_s - T_s x \\ & y \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2} \end{aligned} \tag{8}$$

Problem (1) can be reformulated as the following Benders model:

$$\min_{x, \theta_s} \left\{ c^T x + \sum_{s \in S} p_s \theta_s : (x, \theta_s) \in E^s, s \in S \right\},$$

where for each $s \in S, E^s$ contains the first-stage constraints and the epigraph of Q_s , i.e.,

$$E^s = \{(x, \theta_s) \in X \times \mathbb{R} : Ax \geq b, \theta_s \geq Q_s(x)\}.$$

In a typical approach to solve (1), each recourse function Q_s is replaced by a cutting-plane underestimate $\hat{Q}_s$ which creates a relaxation and is dynamically updated. In each iteration, an approximate problem defined by the current underestimate is solved to obtain a candidate solution and then a cut generation problem is solved to update the cutting plane approximation if necessary. This process is repeated until no cuts are identified.

We review two types of valid inequalities for E^s . The first collection of cuts are Benders cuts. Benders cuts are generated based on the LP relaxation of the problem (2) defining $Q_s(x)$. Given a candidate solution $\hat{x}$, the LP relaxation of the recourse problem is solved:

$$\min_{y} \left\{ (q^s)^T y : W^s y \geq h^s - T^s \hat{x} \right\}. \tag{9}$$

Let μ be an optimal dual solution of problem (3). Based on LP duality, the following Benders cut is valid for E^s :

$$\mu^T T^s x + \theta_s \geq \mu^T h^s. \tag{10}$$

If the SIP has continuous recourse, i.e., $p_2 = 0$, then (4) is tight at $\hat{x}$, i.e., $Q_s(\hat{x}) = -\mu^T T^s \hat{x} + \mu^T h^s$. The cutting-plane model $\hat{Q}_s$ is often constructed by iteratively adding Benders cuts until the lower bound converges to the LP relaxation bound. Benders cuts are sufficient to provide convergence for solving SIPs with continuous recourse.

Another useful family of cuts is the integer L-shaped cuts introduced in [17]. These cuts are valid only when the first-stage variables are binary, i.e., $X = \{0,1\}^n$, but can be applied even when the second-stage includes integer decision variables. In this case, given $\hat{x} \in \{0,1\}^n$ and a value L_s such that $Q_s(x) \geq L_s$ for all feasible x , the following integer L-shaped cut is a valid inequality for E^s :

$$\theta_s \geq Q_s(\hat{x}) - (Q_s(\hat{x}) - L_s)\left(\sum_{i:\hat{x}_i=1}(1-x_i) + \sum_{i:\hat{x}_i=0} x_i\right). \tag{11}$$

The branch-and-cut algorithm can be implemented using a lazy constraint callback in modern MIP solvers, which allows the addition of Benders or integer L-shaped cuts when the solver encounters a solution $(\hat{\theta}, \hat{x})$ with $\hat{x} \in X$ (i.e., $\hat{x}$ satisfies any integrality constraints) but for which $\hat{\theta}_s < Q_s(\hat{x})$ for some $s \in S$. These two classes of cuts are sufficient to guarantee convergence for SIPs with continuous recourse or pure binary first-stage variables. However, the efficiency of the algorithm depends significantly on the strength of the cutting-plane models $\hat{Q}_s$'s. Given poor relaxations of the recourse functions, the branch-and-bound search may end up exploring a huge number of nodes, resulting in a long solution time. As discussed in Section 1 many methods have been proposed to strengthen the model $\hat{Q}_s$ in order to accelerate the algorithm.

We note that the validity of Lagrangian cuts does not require either continuous recourse or pure binary first-stage variables. However, outside of those settings, additional cuts or a specialized branching scheme would be required to obtain a convergent algorithm. I refer the readers to, e.g., [22], [12] for examples of methods that can be used to obtain a finitely convergent algorithm in other settings. Lagrangian cuts could potentially be added to enhance any of these approaches.

4 Preliminaries on Non-linear Optimization

In this section we summarize some basic facts about nonlinear programming problems written in the standard form

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } g_i(x) \leq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{12}$$

Throughout this section, we assume that for $f, g_i : \mathbb{R}^n \longrightarrow \mathbb{R}$, at least one of them is not a linear function and all of them are continuously differentiable. This implies that any local minimum of program (1) is a global minimum.

First of all, we have the two following well known facts.

Proposition 4.1. *The function $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}$ is convex iff for all arbitrarily chosen $x, y \in \mathbb{R}^n$ we have $(y-x)^{\mathrm{T}} \nabla\varphi(x) \leq \varphi(y) - \varphi(x)$.*

Lemma 4.2. *(Farkas' lemma)*
$\{x \mid Ax = b, x \geq 0\} \neq \emptyset$ if and only if $A^{\mathrm{T}} u \geq 0$ implies that $b^{\mathrm{T}} u \geq 0$

Then we present some regularity conditions and corresponding propositions.

Proposition 4.3. *Let $I(\hat{x}) := \{i \mid g_i(\hat{x}) = 0\}$ and $\mathcal{B} = \{x \mid g_i(x) \leq 0, i = 1, \cdots, m\}$ be the feasible region of program (1). Here we present some regularity conditions.*

Kuhn–Tucker condition : $\exists \hat{u} \geq 0$ *such that* $\nabla f(\hat{x}) + \sum_{i=1}^m \hat{u}_i \nabla g_i(\hat{x}) = 0$, $\sum_{i=1}^m \hat{u}_i g_i(\hat{x}) = 0$.

$\mathcal{RC}\_0 : z^{\mathrm{T}} \nabla g_i(\hat{x}) \leq 0, i \in I(\hat{x})$ *implies that* $z^{\mathrm{T}} \nabla f(\hat{x}) \geq 0$.

$\mathcal{RC}\_1$: $\forall z \neq 0$ *s.t.* $z^{\mathrm{T}} \nabla g_i(\hat{x}) \leq 0, i \in I(\hat{x}), \exists \{x^k \mid x^k \neq \hat{x}, k = 1, 2, \cdots\} \subset \mathcal{B}$ *such that* $\lim_{k\to\infty} x^k =$

$\hat{x}$, $\lim_{k\to\infty} \frac{x^k-\hat{x}}{\|x^k-\hat{x}\|} = \frac{z}{\|z\|}$.

$\underline{\mathcal{RC}\_2}$: $\exists \tilde{x} \in \mathcal{B}$ *such that* $g_i(\tilde{x}) < 0, \forall i$.

Then we have some relevant properties

(1) $\underline{\mathcal{RC}\_2} \implies \underline{\mathcal{RC}\_1} \implies \underline{\mathcal{RC}\_0}$ *where the first implication requires the convex condition.*

(2) If $\hat{x}$ *(locally) solves program (1) and satisfies* $\underline{\mathcal{RC}\_0}$ *then the Kuhn-Tucker condition necessarily hold in* $\hat{x}$.

(3) If we assume the program is convex and $\underline{\mathcal{RC}\_2}$ *holds, then* $\hat{x} \in \mathcal{B}$ *(globally) solves problem (1) if and only if the Kuhn-Tucker condition is satisfied in* $\hat{x}$.

Proof. (2) is a consequence of Farkas' Lemma. Others are omitted. Interested readers may refer to [34]. □

When solving stochastic programs, we somtimes need to use preliminary results from both linear and nonlinear programming, and their underlying ideas. Unlike linear programs, nonlinear programs generally cannot be solved in finitely many steps. Instead, we shall have to deal with iterative procedures that we might expect to converge, to some extent, to a solution of the nonlinear program under consideration. Traditional methods contain cutting-plane methods, methods of descent, penalty methods and Lagrangian methods. We will present one particular variant of the two methods.

4.1 Descent Method

The feasible direction and the reduced gradient methods have been extended to the case of nonlinear constraints. However, for the sake of simplicity, we consider the special case of minimizing a convex function under linear constraints. Assume that we have a feasible point $z \in \mathcal{B} = \{x \mid Ax = b, x \geq 0\}$. Then there are two possibilities as we will discuss below.

If z is optimal then the Kuhn-Tucker conditions have to hold. These are, with $J(z) := \{j \mid z_j > 0\}$,

$$\begin{aligned} A^{\mathrm{T}}u - w &= -\nabla f(z) \\ w_j &= 0 \text{ for } j \in J(z), \\ w &\geq 0 \end{aligned}$$

Applying Farkas' Lemma, we have this system is feasible if and only if

$$[\nabla f(z)]^{\mathrm{T}} d \geq 0 \quad \forall d \in \{d \mid Ad = 0, d_j \geq 0 \text{ for } j \notin J(z)\}\,.$$

If the feasible point z is not optimal then the Kuhn-Tucker conditions cannot hold. Therefore, there exists a direction d such that $Ad = 0, d_j \geq 0\ \forall j : z_j = 0$ and $[\nabla f(z)]^{\mathrm{T}} d < 0$. A direction like this is called a feasible descent direction at z , which has to satisfy the following two conditions: $\exists \lambda_0 > 0$ such that $z + \lambda d \in \mathcal{B}$ $\forall \lambda \in [0, \lambda_0]$ and $[\nabla f(z)]^{\mathrm{T}} d < 0$. Hence, having at a feasible point z a feasible descent direction d (for which, by its definition, $d \neq 0$ is obvious), it is possible to move from z in direction d with some positive step length without leaving $\mathcal{B}$ and at the same time at least locally to decrease the objective's value. From these brief considerations, we may state the following.

Algorithm 4.4. (descent directions)

$\underline{\text{Step1}}$: Determine a feasible solution $z^{(0)}$, let $k := 0$.

$\underline{\text{Step2}}$: If there is no feasible descent direction at $z^{(k)}$ then stop $\left(z^{(k)} \text{ is optimal}\right)$. Otherwise, choose a feasible descent direction $d^{(k)}$ at $z^{(k)}$ and go to step 3.

$\underline{\text{Step3}}$: Solve the so-called line search problem $\min_\lambda \left\{ f\left(z^{(k)} + \lambda d^{(k)}\right) \mid \left(z^{(k)} + \lambda d^{(k)}\right) \in \mathcal{B} \right\}$ and with its solution λ_k define $z^{(k+1)} := z^{(k)} + \lambda_k d^{(k)}$. Let $k := k + 1$ and return to step 2 .

We have two famous algorithms for determining $d^{(k)}$: the feasible direction method and the reduced gradient method. Interested readers may refer to classical nonlinear textbooks.

4.2 Penalty Method

The term "penalty" reflects the following attempt. Replace the original program (1) by appropriate free , or unconstrained, optimization problems

$$\min_{x\in\mathbb{R}^n} F_{rs}(x) := f(x) + r \sum_{i\in I} \varphi\left(g_i(x)\right) + \frac{1}{s} \sum_{i\in J} \psi\left(g_i(x)\right) \tag{13}$$

where $I, J \subset \{1, \cdots, m\}$ such that $I \cap J = \emptyset, I \cup J = \{1, \cdots, m\}$, and the parameters $r, s > 0$ are to be chosen or adapted in the course of the procedure. The role of the functions φ and ψ is to inhibit and to penalize respectively the violation of any one of the constraints. More precisely, for these functions we assume that: φ, ψ are monotonically increasing and convex; the so-called barrier function satisfies

$$\begin{aligned} \varphi(\eta) &< +\infty \quad \forall \eta < 0, \\ \lim_{\eta \uparrow 0} \varphi(\eta) &= +\infty \end{aligned}$$

and for the so-called loss function we have

$$\psi(\eta) \begin{cases} = 0 & \forall \eta \leq 0 \\ > 0 & \forall \eta > 0 \end{cases}$$

5 Preliminaries on Gröbner Basis and Graver Basis

Assume A to be an integer matrix, we further study the relationship between Test Set and (Reduced) Gröbner Basis of

$$(IP)_{c,b} : \min \{cx : Ax = b \text{ and } x \in N^n\}$$

First we need to transfer $IP_{c,b}$ to $IP_{>_c,b}$ where $>_c$ denotes the complete total order.

Definition 5.1. $>_c$ is the complete total order satistifying: $x >_c$ y if
1.$c \cdot x > c \cdot y$ or
2. $c \cdot x > c \cdot y$ and $x > y$ where $>$ is an arbitrarily assigned monomial order.

Note that $>_c$ satisfies (Attention: it's not a term order but almost is!)
1. $>_c$ is a total order on N^n.
2. $>_c$ is cmpatible with sum.

Therefore we may make use of the refinement so as to get to the unique optimum.

Lemma 5.2. $\exists \alpha_i$ *s.t.* $\{$*non-optimal solutions of all fibres*$\} = \bigcup_{i=1}^{t} (\alpha(i) + N^n)$ *(immediate consequence of Gordan Dickson Lemma*[**?**]*)*

Theorem 5.3. $\exists$ *testing set for* $(IP)_{c,b}$.

Proof. (Geometric) First we construct $\mathcal{G}_A = \{(\alpha(i) - \beta(i)), i = 1, 2, ..., t\}$ where $\alpha(i)$ is given below and $\beta(i)$ is the corresponding unique optimum point of $(IP)_{>_c,b}(A\alpha(i))$. Then we apply lemma 6.2.

We can therefore draw an arrow from $\alpha(i)$ to $\beta(i)$ and translate it to all feasible points in $IP_{\{A,c\}}$ such that the translated vector is incident at the corresponding feasible points. By this construction we get to a connected digraph with the unique sink at the optimum point.

It's an easy corollary that $\mathcal{G}_A$ is a test set. □

How does the name birth? We need to look at the map below and theorem below explains everything. (We denote $A = [A_1, A_2, ..., A_n]$ by column blocking and y^{A_1} short for a monomial of $k[y_1, y_2, ..., y_m]$ where m is the column number of A)

$$\begin{aligned} \pi : k[x_1, x_2, ..., x_n] &\to k[y^{A_1}, y^{A_2}, ..., y^{A_n}] \\ x_i &\rightarrowtail y^{A_i} \end{aligned}$$

First we define a $\mathbb{Z}$-linear mapping

$$\begin{aligned} \pi_* : \mathbb{Z}^n &\to \mathbb{Z}^m \\ u &\rightarrowtail Au \end{aligned}$$

Then π and π_* are related as $\pi(x^u) = \pi(x_1^{u_1} x_2^{u_2} ... x_n^{u_n}) = y^{A_1 u_1} ... y^{A_n u_n} = y^{Au} = y^{\pi_*(u)}$. we call the kernel of π as **toric ideal** of A, denoted by I_A.

Lemma 5.4. *The toric ideal I_A is spanned as k-vector space by the set of binomials*

$$J = \{x^u - x^v \mid u, v \in \mathbb{N}^n \text{ with } \pi_*(u) = \pi_*(v)\}.$$

Proof.A binomial $x^u - x^v$ lies in I_A if and only if $\pi(x^u - x^v) = \pi(x^u) - \pi(x^v) = x^{\pi_*(u)} - x^{\pi_*(v)} = 0$. What remains to show is that every polynomial $f \in I_A$ is a linear combination of such binomials with coefficients in k . Fix a term order $<$ on $k[x_1, \ldots, x_n]$. Suppose $f \in I_A$ can not be written as such a linear combination. Choose f such that $LM_<(f) = x^u$ is smallest with respect to $<$ for all such polynomials. Since $f \in I_A$ we know

$$0 = \pi(f) = f\left(y^{A_1} \cdots y^{A_n}\right) = y^{\pi_*(u)} + \text{ other terms.}$$

In particular, we know that the term $y^{\pi_*(u)}$ must cancel. Therefore, there exists a monomial x^v in f , with $x^u > x^v$ such that $\pi(u) = \pi(v)$. We know that $f' = f - (x^u - x^v)$ cannot be written as a k-linear combination of binomials since otherwise f could. Since now $LM(f) < LM(f')$, we come to a contradiction for the minimality property of f. The lemma is then proved. □

Theorem 5.5. $I_A = Ker(\pi) =< (x^{\alpha(i)} - x^{\beta(i)}), i = 1, 2, ..., s >$. *And actually* $\{x^{\alpha(i)} - x^{\beta(i)}, i = 1, 2, ..., s\}$ *forms a reduced Grobner basis.*

Proof. Define

$$\phi(u) := x^{u^+} - x^{u^-}$$

Then we carry out to prove the theorem. First, it's trivial that $x^{\alpha_i} - x^{\beta_i} \in Ker(\pi_*)$ since $A\alpha_i = A\beta_i$. Since $\alpha_i >_c \beta_i$ for all $i = 1, ..., s$, we have that

$$LM_{>_c}(x^{\alpha_i} - x^{\beta_i}) = x^{\alpha_i} \implies \left\langle x^{\alpha_i} \right\rangle \subset LM_{>_c}(Ker(\pi)).$$

From the lemma it is enough to show that $x^\alpha \in \left\langle x^{\alpha(i)}, i = 1, \ldots, s \right\rangle$ for each binomial $x^\alpha - x^\beta \in J$. We may assume that $LM_>(x^\alpha - x^\beta) = x^\alpha$. Now $LM_>\left(x^\alpha - x^\beta\right) = x^\alpha$ implies that $\alpha >_c \beta$. Therefore α is a nonoptimal point with respect to $>_c$ in the $A\alpha$-fiber of IP. Therefore α is in the set of all nonoptimal points from all fibers of IP, which is $\bigcup_{i=1}^{s} (\alpha(i) + \mathbb{N}^n)$. This implies that $\alpha = \alpha(i) + v$ for some $i \in \{1, \ldots, s\}$ and $v \in \mathbb{N}^n$. Therefore $x^{\alpha(i)}$ divides x^α which in turn implies that $x^\alpha \in \left\langle x^{\alpha(i)}, i = 1, \ldots, s \right\rangle$. Therefore $\{x^{\alpha(i)} - x^{\beta(i)}, i = 1, 2, ..., s\}$ is a Gröbner basis for I_A with respect to $>_c$, which is clearly reduced observing from their construction. □

We give a general method for specifying monomial orders on $k[x_1, ..., x_n]$. We assert the following lemma without proof since it's useful to prove the theorem below.

Lemma 5.6. *Given any $m \times n$ real matrix M and an monomial order $>$. Then we define $x^\alpha >_M x^\beta$ if and only if*

$$M \cdot \alpha > M \cdot \beta.$$

Let M be an $m \times n$ real matrix with non-negative entries s.t. $ker(M) \cap \mathbb{Z}^n = \{0\}$.Then $>_M$ is a monomial order on $k[x_1, ..., x_n]$. (See Exercise 8 of §2 of [?].)

Theorem 5.7. *(optional/new) By giving different term order $>$ for $>_c$, we can use the $>_c$ and the corresponding geometric bunchberger algorithm to get all optimum points of the problem $IP_{c,b}$. (Recall that we should specify the composite order to use the geometric bunchberger's algorithm.)*

Proof. Suppose the optimum of the $(IP)_{c,b}$ forms a set $\{x_1, x_2, ..., x_m\}$. Then it's clear $\forall j\ \exists c > 0 (c \cdot x_j \neq c \cdot x_i, \forall i \neq j)$. Actually we can prove it by contradiction: since $\sum_i m(\{c : c \cdot x_j = c \cdot x_i\})=0$, there must exist some c such that $c \cdot x_j \neq c \cdot x_i, \forall i \neq j$. ($m()$ denotes the Lebesgue measure.)

Therefore we can find the interval $(a_j, \tilde{a}_j] \subset \mathbb{R}$ s.t.

1. $c \cdot x_i \in (a_i, \tilde{a}_i]$, $\forall i$.
2. $(a_j, \tilde{a}_j] \cap (a_i, \tilde{a}_i] = \emptyset, \forall i \neq j$.

Then we define a total order on $\mathbb{R}$, denoted by $>_s$ s.t.

1. Define $A = \left(\mathbb{R} - \bigcup_{i \neq j} (a_i, \tilde{a}_i]\right)$, $B = \bigcup_{i \neq j} (a_i, \tilde{a}_i]$, and $C = (a_j, \tilde{a}_j]$.
2. All the elements in A, B, C are compared in the usual way.
3. If $a \in A, b \in B, c \in C$, a and b are compared in the usual way; $b < c$; $a < c$ iff. $\exists d \in B (a \leq d)$.

Therefore we get to a total order on $\mathbb{R}$ such that $b <_s c$ whenever $b \in B$ and $c \in C$. Applying the order $>_s$ to build up the monomial order on $k[x_1, ..., x_n]$ as follows. We let

$$M = \begin{pmatrix} c^T \\ ... \end{pmatrix}$$

where ... is filled with positive numbers such that M satisfies the conditions of lemma 3.

Finally we can form the composite order $>_c$ of cost and $>_M$, which satisfies $x_j >_c x_i, \forall j \neq i$. □

5.1 New Algorithms in IP

Recently, various algebraic integer programming (IP) solvers have been proposed based on the theory of Gröbner bases. The main difficulty of these solvers is the size of the Gröbner bases generated. So we propose an algorithm calculating the test set of $(IP)_{c,b}$ much faster.

Recall that

$$\pi : k[x_1, x_2, ..., x_n] \to k[y^{A_1}, y^{A_2}, ..., y^{A_n}]$$
$$x_i \rightarrowtail y^{A_i}$$

and

$$\pi_* : \mathbb{Z}^n \to \mathbb{Z}^m$$
$$u \rightarrowtail Au$$

From the above section, we know that the algorithm in Gröbner bases of IP is to find a "minimal" generator of $Ker(\pi)$, i.e. $Ker(\pi_*)$.

From [**?**], we have an important observation. First we recall the definition

$$\phi(u) := x^{u^+} - x^{u^-}.$$

Theorem 5.8. *Let $K \in \mathbb{N}^{k \times n}$. Then $\phi(K) = \phi(span(K))$*

We let K be a basis for $Ker(\pi_*)$ consisting of k elements (to simplify notation, we will use K to denote a basis for $Ker(\pi_*)$ as well as the matrix in $\mathbb{Z}^{k \times n}$ whose rows are the vectors in K).

Theorem 5.9. *For $K, K' \in \mathbb{Z}^{k \times n}$, define $K' \sim K$ if $spanK' = spanK$, i.e. $K' = AK$ for some $A \in \mathbb{Z}^{k \times k}$ s.t. $|det(A)| = 1$.*

From definition we immediately get

Proposition 5.10. *Let $K \in \mathbb{Z}^{k \times n}$. Then there exists a $\tilde{K} \sim K$ such that each column vector of $\tilde{K}$ is either in $\mathbb{N}^k$ or $(-\mathbb{N})^k$.*

From Proposition 3, we get to $\tilde{K}$. Then let $J \subset \{1, 2, ..., n\}$ be the index set of all columns with negative entries, and let K' be the matrix obtained from $\tilde{K}$ by reversing all signs in the columns indexed by J.

Here's another important observation. First we define $T_j : \mathbb{Z}^n \to \mathbb{Z}^n$ as the operator that switches the sign of the j-th component of the vectors in $\mathbb{Z}^n$. Further, if $p \in k[x_1, ..., x_n]$ has the form $p = \phi(u)$ for some $u \in \mathbb{Z}^n$, we denote $T_j(p) = \phi(T_j(u))$.

Theorem 5.11. *Let $K \in \mathbb{Z}^{k \times n}$ and assume that there exists a finite set $U \subset \operatorname{span} K$ such that $\langle \varphi(U) \rangle = \langle \varphi(\operatorname{span} K) \rangle$. If G is the reduced Gröbner basis for $\langle \varphi(U) \rangle$, with respect to a term order that eliminates x_j , then $\langle T_j G \rangle = \langle \varphi(\operatorname{span}(T_j K)) \rangle$.*

We are now ready to describe our algorithm to calculate $\ker \pi$. Let K be a basis for $\ker \pi_*$. By Lemma 3.8 there exists an equivalent basis K' such that each column of K' is either in $\mathbb{N}^n$ or in $(-\mathbb{N})^n$. Let $J \subseteq \{1, 2, \ldots, n\}$ be the index set of all columns with negative entries, and let K'_J be the matrix obtained from K' by reversing all signs in the columns indexed by J. By Theorem 2,

$$\langle \varphi(K'_J) \rangle = \langle \varphi(\operatorname{span} K'_J) \rangle .$$

If $J = \varnothing$ we are done. If $J \neq \varnothing$, let j be any element of J. Theorem 3 enables us to derive from $\varphi(K'_J)$ a finite set of generators for $\left\langle \varphi\left(\operatorname{span} K'_{J\setminus\{j\}}\right)\right\rangle$. Compute the Gröbner basis for $\varphi(K'_J)$ with respect to a term order that eliminates x_j and apply the operator T_j to it. Proceeding recursively, we can calculate a finite set of generators for $\varphi(\operatorname{span} K'_J)$, which by Theorem 2 equals $\ker \pi_*$.

Proposition 5.12. *(new) The proposed algorithm requires the determination of at most* $[\frac{1}{2}n]$ *Grobner bases over* $k[x_1, x_2, ..., x_n]$.

Proof. Actually the number of determination is at most the number of the elements of J.

Remark: based on the (empirical) fact that the complexity of the Buchberger algorithm is a strongly growing function of the number of variables, we conclude that it's in general more efficient to evaluate $[\frac{1}{2}n]$ Grobner bases over $k[x_1, x_2, ..., x_n]$ than one Grobner basis over $k[x_1, x_2, ..., x_n, y_1, y_2, ..., y_m]$. □

The algorithm in P392 of [**?**] is based on the Elimination lemma applied to the $k[x_1, x_2, .., x_n, y_1, y_2, ..., y_m]$, which is **not applicable in large scale**. And so is the Geometric Buchberger's algorithm[1]. So from literature [4], we give a new idea of algorithm below

Finding Reduced Grobener Basis → Augmentation Algorithm

The detailed algorithm to solve the IP is gathered as follows. The value of the algorithm is that it is more efficient to calculate a moderate number of Gröbner bases over $k[x_1, ..., x_n]$ instead of over $K[x_1, ..., x_n, y_1, ..., y_m]$. This is especially true for the memory requirements of the proposed algorithm.

Algorithm 5.13. (new) computation of reduced Gröbner bases
1. Calculate a basis K for $\ker \pi_*$.
2. Find an equivalent basis K' such that all rows of K' lie in the same orthant.
3. Let J be the index set of all columns with negative entries and let K'_J be the matrix obtained from K' by reversing the signs of the columns indexed by J.
4. Let $G_J = \varphi(K'_J)$.
5. Until $J = \varnothing$, repeat this: Take $j \in J$ and let $G_{J\setminus\{j\}}$ be the result of T_j operating on the reduced Gröbner basis for $< G_J >$ with respect to a term order that eliminates x_j ; then let $J \leftarrow J\setminus\{j\}$.
6. Output $G_\varnothing$, a generating set for $\ker \pi$ which is finite.
7. Use $G_\varnothing$ to generate the reduced grobner Basis $\{x^{\alpha(i)} - x^{\beta(i)} : i = 1, 2..., s\}$. (e.g. apply BunchBerger Algorithm)
8. Augmentation Algorithm.

6 Lagrangian Relaxation Based Method

The idea of the following method is to create copies of the first-stage decision variables for each scenario and then apply Lagrangian dualization to decompose the SIP. Recall that the Lagrangian dual of an LP problem simply generalizes the LP dual. ([4])

First we recall the Deterministic equivalent of an SMIP is

$$\begin{aligned} \min \quad & c^T x + \textstyle\sum_{i=1}^{N} p_i q_i^T y_i \\ \text{s.t.} \quad & Ax \geq b \\ & T_i x + W_i y_i = h_i \\ & x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \\ & y_i \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \quad i = 1, \ldots, N \end{aligned} \tag{14}$$

By adding the nonanticipativity constraints $x_i = \sum_{i'=1}^{N} p_{i'} x_{i'}$, we have the equivalent form

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{N} p_i (c^T x_i + q_i^T y_i) \\ \text{s.t.} \quad & Ax_i \geq b, \quad i = 1, \ldots, N \\ & T_i x_i + W_i y_i = h_i, \quad i = 1, \ldots, N \\ & x_i = \textstyle\sum_{i'=1}^{N} p_{i'} x_{i'}, \quad i = 1, \ldots, N \\ & x_i \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, \quad i = 1, \ldots, N \\ & y_i \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \quad i = 1, \ldots, N \end{aligned} \tag{15}$$

Then we relax these constraints using Lagrangian Relaxation with dual vectors $\lambda = (\lambda_1, ..., \lambda_N)$:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{N} p_i (c^T x_i + q_i^T y_i) + \sum_{i=1}^{N} p_i \lambda_i^T (x_i - \sum_{i'=1}^{N} p_{i'} x_{i'}) = \sum_{i=1}^{N} p_i ((c + \lambda_i - \bar{\lambda})^T x_i + q_i^T y_i) \\ \text{s.t.} \quad & Ax_i \geq b, \quad i = 1, \ldots, N \\ & T_i x_i + W_i y_i = h_i, \quad i = 1, \ldots, N \\ & x_i \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, \quad i = 1, \ldots, N \\ & y_i \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \quad i = 1, \ldots, N \end{aligned} \tag{16}$$

where we rewrite the objective by $\bar{\lambda} = \sum_{i=1}^{N} p_i \lambda_i$.

Therefore, we have the Lagrangian relaxation problem decomposes as $\mathcal{L}(\lambda) = \sum_{i=1}^{N} p_i D_i(\lambda_i)$ where $D_i(\lambda_i)$ is

$$\begin{aligned} \min \quad & (c + \lambda_i - \bar{\lambda})^T x_i + q_i^T y_i \\ \text{s.t.} \quad & Ax_i \geq b \\ & T_i x_i + W_i y_i = h_i \\ & x_i \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \\ & y_i \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2} \end{aligned} \tag{17}$$

Then we define the value of Lagrangian dual $w^{LD} := \max\{\mathcal{L}(\lambda) : \bar{\lambda} = \sum_{i=1}^{N} p_i \lambda_i = 0\}$, and we give a relevant theorem (Readers can also refer to proposition 2 in [**?**] , but the proof in that paper has a gap which is fixed below.)

Theorem 6.1.

$$w^{LD} = \min \left\{ c^T x + \sum_{i=1}^{N} p_i q_i y_i : (x, y_i) \in \operatorname{conv}(X_i), i = 1, \ldots, N \right\}$$

where for $i = 1, \ldots, N$

$$\begin{aligned} X_i := \{(x, y) : & Ax \geq b, T_i x + W_i y = h_i \\ & x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, y \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2} \} \end{aligned} \tag{18}$$

Proof. From Theorem 6.2 in [5], p. 327, we have that

$$\max\{\mathcal{L}(\lambda) : \lambda \geq 0 \text{ and } \bar{\lambda} = \sum_{i=1}^{N} p_i \lambda_i = 0\} = \min \left\{ \sum_{i=1}^{N} p_i (c^T x_i + q_i^T y_i) : x_i \leq \sum_{i'=1}^{N} p_{i'} x_{i'}, (x_i, y_i) \in \operatorname{conv}(X_i), i = 1, \ldots, N \right\}$$

Similarly, we have

$$\max\{\mathcal{L}(\lambda) : \lambda \leq 0 \text{ and } \bar{\lambda} = \sum_{i=1}^{N} p_i \lambda_i = 0\} = \min \left\{ \sum_{i=1}^{N} p_i (c^T x_i + q_i^T y_i) : x_i \geq \sum_{i'=1}^{N} p_{i'} x_{i'}, (x_i, y_i) \in \operatorname{conv}(X_i), i = 1, \ldots, N \right\}$$

From the two equations above, we have the result. □

Here immediately follow some properties.

Proposition 6.2. *Here are some relationships between w^{LD}, z^{SMIP} and z^{SLP}:*
(1) In general $w^{LD} < z^{SMIP}$
(2) $w^{LD} \geq z^{SLP}$ (the usual LP relaxation)
(3) w^{LD} at least as good as any bound obtained using cuts in single scenario sub-problems
(4) In many test instances, w^{LD} is very close to z^{SMIP} .

Therefore, it suffices to search for w^{LD} where $w^{LD} = \max\{\mathcal{L}(\lambda) : \bar{\lambda} = \sum_{i=1}^{N} p_i \lambda_i = 0\}$ is the optimization program of a piece-wise linear and convex function.

The method can be generalized to the case of a multi-stage stochastic program. Interested readers can go to [3].

6.1 Lagrangian Relaxation Based Method with Graver Basis

We give without proof the theorem below. Suppose

$$A_N := \begin{pmatrix} W_1 & 0 & \cdots & 0 \\ 0 & W_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_N \end{pmatrix}$$

Theorem 6.3. *Let $\mathcal{GR}(\cdot)$ denotes the Graver Basis of a matrix, then $\mathcal{GR}(A_N) = \{(0, 0, .., v_i, ..., 0)$ where $v_i \in \mathcal{GR}(W_i)\}$. Similar results hold for Grobner Basis.*

Lagrangian duality provides lower bounds on the optimal value of problem and corresponding optimal solutions $\left(x^j, y^j\right), j = 1, \ldots, N$, of the Lagrangian relaxation. In general, these scenario solutions will not coincide in their x-component unless the duality gap vanishes. We now elaborate a branch and bound procedure for Eq. (1) that uses Lagrangian relaxation of non-anticipativity constraints as bounding procedure. To come up with candidates for feasible first-stage solutions x various heuristic ideas starting from the scenario solutions $x^j, j = 1, \ldots, N$ can be tried. In the present paper we use the average $\bar{x} = \sum_{j=1}^{r} p^j x^j$, combined with some rounding heuristic in order to fulfill the integrality restrictions. In the following, P denotes the list of current problems together with associated lower bounds $z_{\mathrm{LD}} = z_{\mathrm{LD}}(P)$. The outline of the algorithm is as follows:

Algorithm 6.4. (Graver Basis, Sub-gradient, Branch and Bound)
Step 1: Compute the Graver Basis of the matrix of the Lagrangian dual of the original problem
Step 2: Initialization: Set $\underline{z} = \infty$ and let P consist of the original problem.
Step 3: Termination: If $P = \emptyset$ then the solution $\hat{x}$ that yielded $\underline{z} = c\hat{x} + Q(\hat{x})$ is optimal.
Step 4: Node selection: Select and delete a problem P from P , solve the corresponding Lagrangian dual (**For each λ, use Graver Basis to compute the corresponding optimum**)whose optimal value yields the bound $z_{\mathrm{LD}} = z_{\mathrm{LD}}(P)$. If P is infeasible ($z_{\mathrm{LD}} = \infty$) go to Step 2 .
Step 5: Bounding: If $z_{\mathrm{LD}}(P) \geqslant \underline{z}$ go to Step 2 (this step can be carried out as soon as the value of the Lagrangian dual falls below $\underline{z}$).

(i) The scenario solutions $x^j, j = 1, \ldots, N$, are identical: Let $\underline{z} := \min\left\{\underline{z}, cx^j + Q\left(x^j\right)\right\}$ and delete from P all problems P' with $z_{\mathrm{LD}}\left(P'\right) \geqslant \underline{z}$. Go to Step 2 .

(ii) The scenario solutions $x^j, j = 1, \ldots, N$ differ: Compute the average $\bar{x}$ and round it by some heuristic to obtain $\bar{x}^{\mathrm{R}}$. If $\bar{x}^{\mathrm{R}}$ is feasible then let $\underline{z} := \min\left\{\underline{z}, c\bar{x}^{-\mathrm{R}} + Q\left(\bar{x}^{\mathrm{R}}\right)\right\}$ and delete from P all problems P' with $z_{\mathrm{LD}}\left(P'\right) \leqslant \underline{z}$. Go to Step 5 .
Step 6: Branching: Select a component x_i of x and add two new problems to P obtained from P by adding the constraints $x_i \leqslant \lfloor \bar{x}_i \rfloor$ and $x_i \geqslant \lfloor \bar{x}_i \rfloor + 1$, respectively (if x_i is an integer component) or $x_i \leqslant \bar{x}_i - \varepsilon$ and $x_i \geqslant \bar{x}_i + \varepsilon$, respectively, where $\varepsilon > 0$ is a tolerance parameter to have disjoint subdomains.

6.2 Lagrangian Cut Based Method

Recall that In dual decomposition for

$$\begin{array}{lll} \min_{x,x^s,y^s} & \sum_{s\in S} p_s \left(c^T x^s + (q^s)^T y^s\right) & \\ \text{s.t.} & Ax^s \geq b, & \\ & T^s x^s + W^s y^s \geq h^s, & s \in S, \\ & x^s \in X, y^s \in Y, & s \in S, \\ & x^s = x, & s \in S \end{array}$$

We have that Lagrangian relaxation is applied to the nonanticipativity constraints $x^s = x$ in this formulation with multipliers λ^s for $s \in S$, which gives the following Lagrangian relaxation problem:

$$\begin{aligned} z(\lambda) = \min_{x,x^s,y^s} & \sum_{s\in S} p_s \left(c^T x^s + (q^s)^T y^s\right) + \sum_{s\in S} p_s \left(\lambda^s\right)^\top (x^s - x) \\ \text{s.t.} \ & (x^s, y^s) \in K^s, \quad s \in S \end{aligned}$$

where $K^s := \{x \in X, y \in Y : Ax \geq b, T^s x + W^s y \geq h^s\}$ for each $s \in S$. Throughout this paper we assume that K^s is nonempty for each $s \in S$.

The BDD algorithm [4] is a version of Benders decomposition that uses strengthened Benders cuts and Lagrangian cuts to tighten the Benders model. In particular, they solve two types of scenario MIPs to generate optimality and feasibility cuts for E^s :

1. Given any $\lambda \in \mathbb{R}^n$, let $(\bar{x}^s_\lambda, \bar{y}^s_\lambda)$ be an optimal solution of

$$\min_{x,y} \left\{\lambda^T x + (q^s)^T y : (x, y) \in K^s\right\}.$$

Then the following Lagrangian optimality cut is valid for E^s :

$$\lambda^T (x - \bar{x}^s_\lambda) + \theta_s \geq (q^s)^T \bar{y}^s_\lambda.$$

2. Given any $\lambda \in \mathbb{R}^n$, let $(\hat{x}^s_\lambda, \hat{y}^s_\lambda, \hat{u}^s_\lambda, \hat{v}^s_\lambda)$ be an optimal solution of

$$\min_{x,y,u,v} \left\{\not{W}^T v + \not{W}^T u + \lambda^T x : Ax + u \geq b, T^s x + W^s y + v \geq h^s, x \in X, y \in Y\right\}.$$

Then the following Lagrangian feasibility cut is valid for E^s :

$$\lambda^T (x - \hat{x}^s_\lambda) \geq \not{W}^T \hat{v}^s_\lambda + \not{W}^T \hat{u}^s_\lambda.$$

The BDD algorithm generates both types of Lagrangian cuts by heuristically solving a Lagrangian cut generation problem. Numerical results from [4] show that Lagrangian cuts are able to close significant gap at the root node for a variety of SIP problems. Our goal in this work is to provide new methods for quickly finding strong Lagrangian cuts. Here to compute every cuts we can apply algorithm with Gröbner Basis and Graver Basis efficiently.

7 Scenario Bundling

This method is based on a simple idea:
(1) Partition scenario set as: $\{1, \ldots, S\} = \bigcup_{k=1}^{K} B_k, B_k \cap B_{k'} = \emptyset$.
(2) When doing decomposition, treat scenarios within each "bundle" as a single scenario.

References

[1] Hemmecke, R., and R. Schultz. "Decomposition of Test Sets in Stochastic Integer Programming." Mathematical Programming, vol. 94, no. 2-3, 2003, pp. 323–341., doi: https://doi.org/10.1007/s10107-002-0322-1.

[2] Schultz, Rüdiger, et al. "Solving Stochastic Programs with Integer Recourse by Enumeration: A Framework Using Gröbner Basis." Mathematical Programming, vol. 83, no. 1-3, 1998, pp. 229–252., doi: https://doi.org/10.1007/bf02680560.

[3] Carøe, Claus C., and Rüdiger Schultz. "Dual Decomposition in Stochastic Integer Programming." Operations Research Letters, vol. 24, no. 1-2, 1999, pp. 37–45., doi: https://doi.org/10.1016/s0167-6377(98)00050-9.

[4] "Integer Programming: Lagrangian Relaxation." SpringerReference, doi: https://doi.org/10.1007/springerreference_72368.

[5] Nemhauser, George L. Integer and Combinatorial Optimization. John Wiley and Sons, 1999.

[6] Benders, J. F. "Partitioning Procedures for Solving Mixed-Variables Programming Problems." Numerische Mathematik, vol. 4, no. 1, 1962, pp. 238–252., doi: https://doi.org/10.1007/bf01386316.

[7] Schultz, Rüdiger. "Continuity Properties of Expectation Functions in Stochastic Integer Programming." Mathematics of Operations Research, vol. 18, no. 3, 1993, pp. 578–589., doi: https://doi.org/10.1287/moor.18.3.578.

[8] Louveaux, François V., and Rüdiger Schultz. "Stochastic Integer Programming." Handbooks in Operations Research and Management Science, 2003, pp. 213–266., doi: https://doi.org/10.1016/s0927-0507(03)10004-7.

[9] Sen, Suvrajeet. "Algorithms for Stochastic Mixed-Integer Programming Models." Discrete Optimization, 2005, pp. 515–558., doi: https://doi.org/10.1016/s0927-0507(05)12009-x.

[10] Ahmed, Shabbir, et al. "A Finite Branch-and-Bound Algorithm for Two-Stage Stochastic Integer Programs." Mathematical Programming, vol. 100, no. 2, 2004, pp. 355–377., doi: https://doi.org/10.1007/s10107-003-0475-6.

[11] Sen, Suvrajeet, and Hanif D. Sherali. "Decomposition with Branch-and-Cut Approaches for Two-Stage Stochastic Mixed-Integer Programming." Mathematical Programming, vol. 106, no. 2, 2005, pp. 203–223., doi: https://doi.org/10.1007/s10107-005-0592-5.

[12] Zhang, Minjiao, and Küçükyavuz Simge. "Finitely Convergent Decomposition Algorithms for Two-Stage Stochastic Pure Integer Programs." SIAM Journal on Optimization, vol. 24, no. 4, 2014, pp. 1933–1951., doi: https://doi.org/10.1137/13092678x.

[13] Sen, Suvrajeet, and Julia L. Higle. "The C3 Theorem and a D2 Algorithm for Large Scale Stochastic Mixed-Integer Programming: Set Convexification." Mathematical Programming, vol. 104, no. 1, 2005, pp. 1–20., doi: https://doi.org/10.1007/s10107-004-0566-z.

[14] Kleywegt, Anton J., et al. "The Sample Average Approximation Method for Stochastic Discrete Optimization." SIAM Journal on Optimization, vol. 12, no. 2, 2002, pp. 479–502., doi: https://doi.org/10.1137/s1052623499363220.

[15] Ntaimo, Lewis. "Disjunctive Decomposition for Two-Stage Stochastic Mixed-Binary Programs with Random Recourse." Operations Research, vol. 58, no. 1, 2010, pp. 229–243., doi: https://doi.org/10.1287/opre.1090.0693.

[16] Zou, Jikai, et al. "Stochastic Dual Dynamic Integer Programming." Mathematical Programming, vol. 175, no. 1-2, 2018, pp. 461–502., doi: https://doi.org/10.1007/s10107-018-1249-5.

[17] Laporte, Gilbert, and François V. Louveaux. "The Integer L-Shaped Method for Stochastic Integer Programs with Complete Recourse." Operations Research Letters, vol. 13, no. 3, 1993, pp. 133–142., doi: https://doi.org/10.1016/0167-6377(93)90002-x.

[18] Carøe, Claus C., and Jørgen Tind. "L-Shaped Decomposition of Two-Stage Stochastic Programs with Integer Recourse." Mathematical Programming, vol. 83, no. 1-3, 1998, pp. 451–464., doi: https://doi.org/10.1007/bf02680570.

[19] Kim, Kibaek, and Sanjay Mehrotra. "A Two-Stage Stochastic Integer Programming Approach to Integrated Staffing and Scheduling with Application to Nurse Management." Operations Research, vol. 63, no. 6, 2015, pp. 1431–1451., doi: https://doi.org/10.1287/opre.2015.1421.

[20] Angulo, Gustavo, et al. "Improving the Integer L-Shaped Method." INFORMS Journal on Computing, vol. 28, no. 3, 2016, pp. 483–499., doi: https://doi.org/10.1287/ijoc.2016.0695.

[21] Laporte, Gilbert, and François V. Louveaux. "The Integer L-Shaped Method for Stochastic Integer Programs with Complete Recourse." Operations Research Letters, vol. 13, no. 3, 1993, pp. 133–142., doi: https://doi.org/10.1016/0167-6377(93)90002-x.

[22] : Ahmed, Shabbir, and Nikolaos V. Sahinidis. "An Approximation Scheme for Stochastic Integer Programs Arising in Capacity Expansion." Operations Research, vol. 51, no. 3, 2003, pp. 461–471., doi: https://doi.org/10.1287/opre.51.3.461.14960.

[23] Gunpinar, Serkan, and Grisselle Centeno. "Stochastic Integer Programming Models for Reducing Wastages and Shortages of Blood Products at Hospitals." Computers Operations Research, vol. 54, 2015, pp. 129–141., doi: https://doi.org/10.1016/j.cor.2014.08.017.

[24] Kim, Kibaek, and Sanjay Mehrotra. "A Two-Stage Stochastic Integer Programming Approach to Integrated Staffing and Scheduling with Application to Nurse Management." Operations Research, vol. 63, no. 6, 2015, pp. 1431–1451., doi: https://doi.org/10.1287/opre.2015.1421.

[25] Wang, Kai, and Alexandre Jacquillat. "A Stochastic Integer Programming Approach to Air Traffic Scheduling and Operations." Operations Research, vol. 68, no. 5, 2020, pp. 1375–1402., doi: https://doi.org/10.1287/opre.2020.1985.

[26] Fischetti, Matteo, and Paolo Toth. "An Additive Bounding Procedure for Combinatorial Optimization Problems." Operations Research, vol. 37, no. 2, 1989, pp. 319–328., doi: https://doi.org/10.1287/opre.37.2.319.

[27] Fischetti, Matteo, and Paolo Toth. "An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem." Mathematical Programming, vol. 53, no. 1-3, 1992, pp. 173–197., doi: https://doi.org/10.1007/bf01585701.

[28] Baldacci, Roberto, et al. "An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation with Additional Cuts." Mathematical Programming, vol. 115, no. 2, 2007, pp. 351–385.,doi: https://doi.org/10.1007/s10107-007-0178-5.

[29] Baldacci, Roberto, and Aristide Mingozzi. "A Unified Exact Method for Solving Different Classes of Vehicle Routing Problems." Mathematical Programming, vol. 120, no. 2, 2008, pp. 347–380., doi: https://doi.org/10.1007/s10107-008-0218-9.

[30] Dupacova, J., et al. "Scenario Reduction in Stochastic Programming." Mathematical Programming, vol. 95, no. 3, 2003, pp. 493–511., doi: https://doi.org/10.1007/s10107-002-0331-0.

[31] Römisch, Werner. "Scenario Reduction Techniques in Stochastic Programming." Stochastic Algorithms: Foundations and Applications, 2009, pp. 1–14., doi: https://doi.org/10.1007/978-3-642-04944-6_1.

[32] Carrion, Miguel, et al. "A Stochastic Programming Approach to Electric Energy Procurement for Large Consumers." IEEE Transactions on Power Systems, vol. 22, no. 2, 2007, pp. 744–754., doi: https://doi.org/10.1109/tpwrs.2007.895164.

[33] Morales, J.M., et al. "Scenario Reduction for Futures Market Trading in Electricity Markets." IEEE Transactions on Power Systems, vol. 24, no. 2, 2009, pp. 878–888., doi: https://doi.org/10.1109/tpwrs.2009.2016072.

[34] Kall, Peter, and Stein W. Wallace. Stochastic Programming. John Wiley Sons, 1997.

[35] Biase, Fausto Di, and Rüdiger Urbanke. “An Algorithm to Calculate the Kernel of Certain Polynomial Ring Homomorphisms.” Experimental Mathematics, vol. 4, no. 3, 1995, pp. 227–234., doi: https://doi.org/10.1080/10586458.1995.10504323.