Integer set reduction for stochastic mixed-integer programming

Saravanan Venkatachalam1 · Lewis Ntaimo2

Received: 10 April 2020 / Accepted: 19 January 2023 / Published online: 9 February 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Two-stage stochastic mixed-integer programs (SMIPs) with general integer variables in the second-stage are generally difficult to solve. This paper develops the theory of integer set reduction for characterizing a subset of the convex hull of feasible integer points of the second-stage subproblem which can be used for solving the SMIP with pure integer recourse. The basic idea is to use the smallest possible subset of the subproblem feasible integer set to generate a valid inequality like Fenchel decomposition cuts with a goal of reducing computation time. An algorithm for obtaining such a subset based on the solution of the subproblem linear programming relaxation is devised and incorporated into a decomposition method for SMIP. To demonstrate the performance of the new integer set reduction methodology, a computational study based on randomly generated knapsack test instances was performed. The results of the study show that integer set reduction aids in speeding up cut generation, leading to better bounds in solving SMIPs with pure integer recourse than using a direct solver.

Keywords Stochastic programming · Integer programming · Integer set reduction · Cutting planes · Fenchel decomposition · Multidimensional knapsack

1 Introduction
A two-stage stochastic mixed-integer programming (SMIP) problem involves optimizing the here-and-now (first-stage) costs plus expected future (second-stage) costs subject to a set of constraints [5]. Solving SMIP is still challenging

---

1 Department of Industrial and Systems Engineering, Wayne State University, 4815 Fourth St, Detroit, MI 48202, USA
2 Wm Michael Barnes ’64 Department of Industrial and Systems Engineering, Texas A & M University, 3131 TAMU, College Station, TX 77843, USA
and this paper makes strides towards that by introducing the theory of integer set reduction for characterizing subsets of the convex hull of feasible integer points of the second-stage (subproblem) that can be used to generate valid inequalities (cutting planes or cuts) for SMIP with pure integer recourse. The goal of integer set reduction is to speed up cut generation routines and potentially lead to faster solution times for SMIP than simply using a direct solver. In this work, we consider SMIP problems with pure integer recourse of the following form:

\[
\text{SIP2: } \begin{align*}
& \text{Max } c^T x + Q_E(x) \\
& \text{s.t. } x \in X.
\end{align*}
\]

In problem SIP2, the set \( X \) is the feasible region of the first-stage, and also includes binary and/or integer restrictions on all or some components of first-stage decision vector \( x \). In other words, the set \( X \) is mixed-integer programming (MIP) representable. The vector \( c \in \mathbb{R}^{n_1} \) is the first-stage cost vector. The function \( Q_E(x) \) is the expected recourse function and it computes the expected second-stage cost based on first-stage decision variable \( x \) and a multivariate random variable \( \tilde{\omega} \). This function is given as follows:

\[
Q_E(x) = \mathbb{E}_{\tilde{\omega}}[\Phi(q(\tilde{\omega}), h(\tilde{\omega}) - T(\tilde{\omega})x, \tilde{\omega})],
\]

where \( q(\tilde{\omega}) \in \mathbb{R}^{n_2} \) is the cost vector, \( h(\tilde{\omega}) \in \mathbb{R}^{m_2} \) is the right hand side vector, and \( T(\tilde{\omega}) \in \mathbb{R}^{m_2 \times n_1} \) is the technology matrix. Letting \( \tau(\omega, x) = h(\omega) - T(\omega)x \) for a given realization \( \omega \) of \( \tilde{\omega} \), the second-stage function \( \Phi \) is a value function of a MIP and is given as follows:

\[
\Phi(q(\omega), \tau(\omega, x), \omega) = \text{Max } \{ q(\omega)^T y(\omega) : Wy(\omega) \leq \tau(\omega, x), 0 \leq y(\omega) \leq u, y(\omega) \in \mathbb{Z}_+^{n_2} \}.
\]

In the second-stage (scenario) subproblem (3), \( y(\omega) \in \mathbb{Z}_+^{n_2} \) is the recourse decision variable vector and \( W \in \mathbb{R}_+^{m_2 \times n_2} \) is the fixed recourse matrix. The vector \( u \in \mathbb{Z}_+^{n_2} \) is the upper bound on the second-stage decision variables. It should also be noted that \( y(\omega) \) depends on \( x \), for convenience the second-stage variable is denoted as \( y(\omega) \) rather than \( y(\omega, x) \).

We consider problem SIP2 under the following assumptions:

(A1) The random variable \( \tilde{\omega} \) is discrete with finitely many scenarios \( \omega \in \Omega \), each with probability of occurrence \( p_\omega \) such that \( \sum_{\omega \in \Omega} p_\omega = 1 \).

(A2) The first-stage feasible set \( X \) is nonempty.

(A3) The second-stage feasible set \( \{ Wy(\omega) \leq \tau(\omega, x), 0 \leq y(\omega) \leq u, y(\omega) \in \mathbb{Z}_+^{n_2} \} \) is bounded and nonempty for all \( x \in X \), and problem (1) has relatively complete recourse, i.e., \( \mathbb{E}_{\omega}[|\Phi(q(\tilde{\omega}), \tau(\tilde{\omega}, x), \tilde{\omega})|] < \infty \) for all \( x \in X \).

(A4) The elements of the recourse matrix \( W \) are non-negative, i.e., \( W \in \mathbb{R}_+^{m_2 \times n_2} \).
Assumption (A1) is needed for tractability, while assumptions (A2) and (A3) are needed to guarantee that the problem has an optimal solution. Also, since $y(\omega)$ has an upper bound $u$, the second-stage feasible set \( \{ Wy(\omega) \leq t(\omega, x), 0 \leq y(\omega) \leq u, y(\omega) \in \mathbb{Z}_+^n \} \) is bounded and nonempty for all $x \in X$. This property is needed for the proposed integer set reduction method to allow for a well-defined problem and finite convergence of the cutting plane method. Because of assumption (A1), SIP2 can be written in an extensive form or deterministic equivalent problem (DEP) as follows:

\[
\text{DEP: Max } \quad c^T x + \sum_{\omega \in \Omega} p_{\omega} q(\omega)^T y(\omega) \\
\text{s.t. } \quad T(\omega)x + Wy(\omega) \leq h(\omega) \quad \forall \omega \in \Omega, \\
\quad x \in X, \quad y(\omega) \in \mathbb{Z}_+^n, \quad \forall \omega \in \Omega.
\]

Even with a reasonable number of scenarios in $\Omega$, DEP is a large-scale MIP. With integer variables in both first- and second-stages, a moderate sized DEP may be difficult to solve using a direct solver such as CPLEX [15] or Gurobi [30]. This makes a decomposition approach necessary for most practical sized problems. In SIP2, the type of decision variables (continuous, binary, integer) and in which stage they appear greatly influences algorithm design. The type of solution method depends on the integral restrictions on both the decision variables $x$ and $y(\omega)$. When both these decision variables do not have integer restrictions, the recourse function $\Phi(q(\omega), t(\omega, x), \omega)$ is a well-behaved piecewise linear and convex function of $x$. Thus, Benders’ decomposition [4] is applicable in this case [47] and the L-shaped method [44] can be used to solve the problem. Hence, the L-shaped method works by approximating the expected recourse function by constructing optimality cuts in the first-stage based on the dual solutions of the subproblems. However, when $y(\omega) \in \mathbb{Z}_+^n$, the L-shaped method is no longer applicable. This is because the value function of an MIP is generally discontinuous and is lower semicontinuous [6]. In addition, the value function is nonconvex and subadditive [34]. Hence, new algorithms are required when there are integer variables in the second-stage, or in both stages.

2 Literature review

In the literature for SMIP, algorithms have been developed with only pure integer variables ([1, 23, 36]), mixed-binary variables ([14, 17, 25, 29, 37, 42]), and mixed-integer variables ([37, 38, 40]) in the second-stage. For comprehensive surveys on SMIP, we refer the reader to ([24, 26, 32, 39]). Cutting plane methods that can partially approximate the second-stage problems within the L-shaped method have been proposed for SMIP with integer variables in the second-stage. A lift-and-project cutting plane approach based on the ideas from [2] is used to solve problems with binary and continuous decision variables in both the first and second-stage [14].

For problems with binary variables in first-stage, and binary and continuous decision variables in second-stage, disjunctive cuts have been developed for the
second-stage [37]. The framework of reformulation linearization technique is used for SIP2 in [41] and [42]. The algorithm in [37] is extended in [38] for problems with binary, continuous and discrete variables in the subproblem.

Fenchel cuts are suggested in [10] and a number of characteristics are derived in [9, 11] and [12]. The most important results from [10, 11] and [12] are that Fenchel cutting planes are facet defining under certain conditions, and the use of Fenchel cuts in a cutting plane approach yields an algorithm with finite convergence. These works also highlight the fact that generating a Fenchel cut for binary programs is computationally expensive in general; therefore, problems with special structure are desirable to achieve faster convergence. Computational experiments demonstrating the effectiveness of Fenchel cuts are presented for knapsack polyhedra in [8] and for pure binary problems in [11].

Since the pioneering work in [10], only a few works in the literature have adopted Fenchel cuts. In [33], Fenchel cuts are used to improve the bounds obtained from MIPs using Lagrangian relaxation. Fenchel cuts are used to solve deterministic capacitated facility location problems [31]. This work compares Fenchel cuts to Lagrangian cuts in finding good relaxation bounds for their problem. In [7], Fenchel cutting planes are used for finding \( p \) median nodes in a graph using a cut-and-branch approach. Fenchel cuts are first derived for two-stage SMIPs under a stage-wise decomposition setting in [28] and are referred to as Fenchel decomposition (FD) cuts. Studies of FD cuts for two-stage SMIP with binary decision variables in both first and second-stages are given in [3] and [45].

Ahmed et al. [1] proposed a branch-and-bound (BAB) algorithm for mixed-integer first-stage decision variables and pure integer decision variables in the second-stage subproblem. Their method transforms the original problem so that discontinuities from the subproblem value function can be handled better. Preliminary computational results were reported and they demonstrate the effectiveness of their approach. A dual decomposition approach for SIP2 was devised in [13]. In this approach, the Lagrangian multipliers associated with the relaxed nonanticipativity constraints are updated from one iteration to the next by a subgradient optimization approach such as the bundle method [22]. Since dual decomposition relies on the Lagrangian dual, it may not be possible to find the (integer) optimal solution to SIP2 because of the existence of the duality gap for the Lagrangian dual of an integer program (IP) [27]. For this reason, the authors implement a BAB procedure to achieve finite convergence to the integer optimal. Another formal treatment of the dual decomposition algorithm is reported [26].

Algorithms based on algebraic methods to exploit the similarities in the second-stage subproblems are proposed in [21] and [36]. In [21], the authors identify the building blocks for SIP2 that can be obtained using computational algebra, and this helps in solving the second-stage integer subproblems efficiently irrespective of number of scenarios. In [36], the first-stage is evaluated by enumeration and bounding, and the second-stage is handled by exploiting the similarities in the integer subproblems. For SIP2 with simple recourse, the work in [18] and [19] define a method to construct the convex envelope of the second-stage value function. Similarly, the work in [43] presents convex approximation approach for the pure integer decision variables in the second-stage, and this provides better lower bounds compared to the
corresponding linear programming (LP) relaxation. Earlier surveys on SIP2 models and algorithms can be found at [20] and [35]. Kong et al. [23] used the superadditivity property of the value function $Q_E(x)$ to devise a BAB algorithm for SIP2.

In this paper, we develop a decomposition algorithm for solving two-stage SMIP with discrete distributions, mixed-integer first-stage variables, and pure-integer second-stage variables. We devise an algebraic method of generating a small finite set of integer points required to generate a valid inequality (Fenchel cuts) for the second-stage subproblem. In addition to the right hand side of the second-stage subproblem, the proposed method allows for uncertainties in the cost parameters, technology matrix, and right-hand side vector in the subproblem. This work makes the following contributions to the literature on stochastic programming: (a) integer set reduction theory for determining subsets of the second-stage feasible integer set to use for faster cut generation; (b) an algorithm for obtaining such subsets based on the solution of the subproblem LP-relaxation; (c) application of the integer set reduction in the context of FD cuts for solving SMIPs with general integer variables in the second-stage; and (d) a computational study that demonstrates the advantages of integer set reduction. In the literature, Fenchel cuts and FD cuts are derived and used for MIP and SMIP, respectively, with binary variables. This work derives these cuts for MIP and SMIP with general integer variables. The derivation of a Fenchel or FD cut requires solving the corresponding second-stage subproblem without removing integrality restrictions. This poses a challenge for second-stage subproblems with integer variables and hence, requires devising a special scheme to generate the cuts.

The rest of this paper is organized as follows: Integer set reduction theory is derived in Sect. 3 and numerical examples are given. The theory is applied to the FD setting for two-stage SMIP in Sect. 4. A computational study to illustrate the application of the new methodology is reported in Sect. 5. Finally, conclusions and directions for future research are given in Sect. 6.

3 Integer set reduction for cut generation

We are now ready to develop integer set reduction theory to characterize the properties of the convex hull of integer points needed for generating a valid inequality for the second-stage feasible set. We use the theory to devise an algorithm for obtaining a reduced set of integer points needed for generating a valid inequality based on the second-stage LP-relaxation. To illustrate the concepts, we use simple numerical examples. Next, we start with the foundations of integer set reduction.

3.1 Foundations of integer set reduction

We begin with some definitions needed in the derivation of a valid inequality for SIP2 with general integer variables in the second-stage. For scenario $\omega$ and first-stage solution $x$, let the feasible set for the second-stage subproblem (3) for $(\omega, x)$ be given as follows:
\[ Y^{IP}(\omega, x) = \{ y(\omega) : Wy(\omega) \leq \tau(\omega, x), 0 \leq y(\omega) \leq u, y(\omega) \in \mathbb{Z}^{n_2}_+ \}. \] (5)

Then its LP-relaxation feasible set is
\[ Y^{LP}(\omega, x) = \{ y(\omega) : Wy(\omega) \leq \tau(\omega, x), 0 \leq y(\omega) \leq u, y(\omega) \in \mathbb{R}^{n_2}_+ \}. \]

Thus, subproblem (3) can now be rewritten as
\[ \Phi(q(\omega), \tau(\omega, x), \omega) = \max \{ q(\omega)^\top y(\omega) : y(\omega) \in Y^{IP}(\omega, x) \} \] (6)
and its LP-relaxation is given as
\[ \Phi_{LP}(q(\omega), \tau(\omega, x), \omega) = \max \{ q(\omega)^\top y(\omega) : y(\omega) \in Y^{LP}(\omega, x) \}. \] (7)

Now let \( y^{LP}(\omega) \in Y^{LP}(\omega, x) \) be the optimal solution to subproblem (7) for a given \( x \in X \). We shall denote by \( \text{conv}(Y^{IP}(\omega, x)) \) the convex hull of integer points in \( Y^{IP}(\omega, x) \). Let \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) be a “small” finite set such that \( \hat{Y}^{IP}(\omega, x, y^{LP}) \subseteq Y^{IP}(\omega, x) \), and \( \text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP})) \) denotes the convex hull of integer points in \( \hat{Y}^{IP}(\omega, x, y^{LP}) \).

Our goal is to use the point \( y^{LP}(\omega) \) and restrict the derivation of valid inequalities (cuts) to a relatively small subset of integer points \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) instead of \( Y^{IP}(\omega, x) \), such that the generated cut that separates the optimal LP solution \( y^{LP}(\omega) \in Y^{LP}(\omega, x) \) from \( \text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP})) \) is also valid for \( \text{conv}(Y^{IP}(\omega, x)) \). By doing so we aim to reduce the computation time for generating a cut, thus enabling a fast cutting plane method for SIP2. Therefore, it is desirable to have \( \hat{Y}^{IP}(\omega, x, y^{LP}) \subseteq Y^{IP}(\omega, x) \) such that \( |\hat{Y}^{IP}(\omega, x, y^{LP})| < |Y^{IP}(\omega, x)| \) so a cut that separates the optimal LP solution \( y^{LP}(\omega) \) using \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) is computationally cheaper than using \( Y^{IP}(\omega, x) \).

**Definition 3.1** An inequality is said to be valid for the set \( \text{conv}(Y^{IP}(\omega, x)) \) if it is satisfied by every point in the set. A cut with respect to a point \( y^{LP}(\omega) \not\in \text{conv}(Y^{IP}(\omega, x)) \) is a valid inequality for \( \text{conv}(Y^{IP}(\omega, x)) \) that is violated by \( y^{LP}(\omega) \).

Generating a valid inequality using the subset \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) is depicted in Fig. 1. In the figure, given \( y^{LP}(\omega) \in Y^{LP}(\omega, x) \), the three points (2,2), (2,3), and (3,2) defining the triangle constitute \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) and are used to generate the cut (dashed line) of the form \( \pi^\top y(\omega) \leq \pi_0 \). We devise a methodology to obtain \( \hat{Y}^{IP}(\omega, x, y^{LP}) \), and subsequently use it to generate a valid inequality that separates the optimal LP solution \( y^{LP}(\omega) \in Y^{LP}(\omega, x) \) from \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) and \( Y^{IP}(\omega, x) \). Also, since \( \hat{Y}^{IP}(\omega, x, y^{LP}) \subseteq Y^{IP}(\omega, x) \), we need to form \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) such that the generated valid inequality does not cut off any integer points in \( Y^{IP}(\omega, x) \).

Let \( I = \{1, \ldots, n_2\} \) be the set of indices of the components of \( y(\omega) \), and \( K \) be the set of indices of the constraints \( Wy(\omega) \leq \tau(\omega, x) \). Also, let the elements of the matrix \( W \) be denoted by \( w_{kt} \), where \( k \in K \) is the constraint index, and \( t \in I \) is the decision variable \( y(\omega) \)'s component index. We also make the following assumption regarding \( W \):

(A5) The polytope defined by \( \text{conv}(Y^{IP}(\omega, x)) \) is full dimensional with dimension \( n_2 \).
Let \( \hat{y}(\omega) \) be any integer point such that \( \hat{y}(\omega) \in Y^{IP}(\omega, x) \). The \( i \)-th component of \( \hat{y}(\omega) \) will be denoted \( \hat{y}_i(\omega) \). In Fig. 1, for example, the integer point \( \hat{y}(\omega) = (3, 2) \) has components \( \hat{y}_1(\omega) = 3 \) and \( \hat{y}_2(\omega) = 2 \). Let \( y^{IP}(\omega) \) be the optimal solution to subproblem (3). Also, define \( \hat{y}(\omega) = \lfloor y^{LP}(\omega) \rfloor \) so that the components are \( \hat{y}_i(\omega) = \lfloor y^{LP}_i(\omega) \rfloor \), for all \( i \in I \). Let \( d_j(\omega, x, \hat{y}(\omega)) \) denote the distance from \( \hat{y}(\omega) \) to the boundary of \( Y^{LP}(\omega, x) \) along the \( j \)-th axis, \( j \in I \). In other words, \( d_j(\omega, x, \hat{y}(\omega)) \) denotes the maximum distance by which \( \hat{y}_j(\omega) \in Y^{IP}(\omega, x) \) can be increased while keeping it feasible to the LP-relaxation when all the components \( i \neq j \) are fixed. Also, let \( u_j \) be the integer upper bound on \( y_j(\omega) \). Define

\[
\begin{align*}
    r^k_j(\omega, x, \hat{y}(\omega)) &:= (\tau_k(\omega, x) - \sum_{t \neq j} w_{kt}\hat{y}_t(\omega))/w_{kj} \quad \forall j \in I, k \in K.
\end{align*}
\]

Then, the distance \( d_j(\omega, x, \hat{y}(\omega)) \) can be calculated as follows:

\[
\begin{align*}
    d_j(\omega, x, \hat{y}(\omega)) &:= \min_{k \in K} \{ r^k_j(\omega, x, \hat{y}(\omega)) - \hat{y}_j(\omega) \}, \quad u_j - \hat{y}_j(\omega) \quad \forall j \in I. \quad (8)
\end{align*}
\]

Observe that in equation (8), the term \( r^k_j(\omega, x, \hat{y}(\omega)) \) represents the distance the \( j \)-th component of \( y(\omega) \) can be increased without violating the \( k \)-th constraint of \( Y^{LP}(\omega, x) \), and subtracting \( \hat{y}_j(\omega) \) from this quantity gives the distance from the selected integer point \( \hat{y}(\omega) \). The value of \( d_j(\omega, x, \hat{y}(\omega)) \) is set to \( u_j - \hat{y}_j(\omega) \) if the upper bound \( u_j \) for \( j \)-th component is more restrictive compared to all the constraints defining the boundary of \( Y^{LP}(\omega, x) \).
Given $Y^{IP}(\omega, x)$, we want to find a set $\hat{Y}^{IP}(\omega, x, y^{LP}) \subseteq Y^{IP}(\omega, x)$ such that every valid inequality for $\text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP}))$ that separates the LP solution $y^{LP}(\omega) \in Y^{LP}(\omega, x)$ is also valid for $\text{conv}(Y^{IP}(\omega, x))$.

**Definition 3.2** Given a fractional subproblem LP solution $y^{LP}(\omega) \in Y^{LP}(\omega, x)$, a valid integer subset $\hat{Y}^{IP}(\omega, x, y^{LP})$ is a set such that any valid inequality for $\text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP}))$ that cuts off $y^{LP}(\omega)$ is also valid for $\text{conv}(Y^{IP}(\omega, x))$.

Observe that given a valid integer subset $\hat{Y}^{IP}(\omega, x, y^{LP})$, then either $\hat{Y}^{IP}(\omega, x, y^{LP}) = Y^{IP}(\omega, x)$ or $\hat{Y}^{IP}(\omega, x, y^{LP}) \subset Y^{IP}(\omega, x)$. For $\hat{Y}^{IP}(\omega, x, y^{LP}) = Y^{IP}(\omega, x)$, it is obvious that entire set $Y^{IP}(\omega, x)$ can be used for generating a valid inequality to cut off a fractional point $y^{LP}(\omega)$. We are therefore, interested in the second case. Specifically, we want to determine a valid integer subset that is as small as possible, which will help us with fast cut generation to cut off the fractional point $y^{LP}(\omega)$. We summarize this result in the following theorem:

**Theorem 3.3** Assuming assumptions (A1)-(A5) hold, given a fractional subproblem LP solution $y^{LP}(\omega) \in Y^{LP}(\omega, x)$, if there is a valid integer subset $\hat{Y}^{IP}(\omega, x, y^{LP}) \subset Y^{IP}(\omega, x)$ then the cardinality of that subset will be at least $n_2$.

**Proof** By assumption (A5), $\text{conv}(Y^{IP}(\omega, x))$ is full dimensional and therefore, $|Y^{IP}(\omega, x)| \geq n_2 + 1$. Thus, to generate a valid inequality for $\text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP}))$ that cuts off $y^{LP}(\omega)$ and is also valid for $\text{conv}(Y^{IP}(\omega, x))$, at least $n_2$ affinely independent integer points are required to construct a facet for $\text{conv}(Y^{IP}(\omega, x))$. Therefore, $|\hat{Y}^{IP}(\omega, x, y^{LP})| \geq n_2$. \hfill \square

Theorem 3.3 establishes the properties of a valid integer subset $\hat{Y}^{IP}(\omega, x, y^{LP})$ needed to generate a valid inequality. Next, we summarize the conditions that $\hat{Y}^{IP}(\omega, x, y^{LP})$ should satisfy so that any valid inequality for $\text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP}))$ that cuts off a fractional point $y^{LP}(\omega)$ is also valid for $\text{conv}(Y^{IP}(\omega, x))$. For any $\hat{y}(\omega) \in \hat{Y}^{IP}(\omega, x, y^{LP})$, one of the following should hold:

(i) $d_j(\omega, x, \hat{y}(\omega)) < 1$, $\forall j \in I$, or
(ii) $d_j(\omega, x, \hat{y}(\omega)) \geq 1$, and for any other (integer) point $\hat{y}'(\omega) \in Y^{IP}(\omega, x)$, such that $\hat{y}'(\omega) \neq \hat{y}(\omega)$, the following applies $\hat{y}'(\omega) - \hat{y}(\omega) \geq 1$, $\forall j \in I$.

When $d_j(\omega, x, \hat{y}(\omega)) < 1$ there does not exist an integer point between $\hat{y}(\omega) \in \hat{Y}^{IP}(\omega, x)$ and the boundary of $Y^{LP}(\omega, x)$ along $j$-th axis. Alternatively, when $d_j(\omega, x, \hat{y}(\omega)) \geq 1$, there exists an integer point $\hat{y}'(\omega) \in Y^{IP}(\omega, x)$ such that $d_j(\omega, x, \hat{y}(\omega)) > d_j(\omega, x, \hat{y}'(\omega))$. This means that there is an integer point $\hat{y}'(\omega)$ between $\hat{y}(\omega)$ and the boundary of $Y^{LP}(\omega, x)$. Since $\hat{y}'(\omega) < \hat{y}(\omega)$ and
\( \hat{y}_j(\omega), \hat{y}'_j(\omega) > 0 \), it implies that \( d_j(\omega, x, \hat{y}(\omega)) - \hat{y}'_j(\omega) > d_j(\omega, x, \hat{y}(\omega)) - \hat{y}_j(\omega) \) and \( \hat{y}'_j(\omega) - \hat{y}_j(\omega) > 0 \). This indicates that \( \hat{y}'_j(\omega) - \hat{y}_j(\omega) \geq 1 \) since \( \hat{y}'_j(\omega), \hat{y}_j(\omega) \in Y^{IP}(\omega, x) \).

However by (ii), if \( d_j(\omega, x, \hat{y}(\omega)) \geq 1 \) and \( \hat{y}'_j(\omega) - \hat{y}_j(\omega) \geq 1, \forall j \in I \), then the point \( \hat{y}(\omega) \in \hat{Y}^{IP}(\omega, x, y^{LP}) \).

Our goal is to devise an algebraic method to construct the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \). Therefore, it is desirable to construct the smallest possible integer subset such that the valid inequality generated does not cut off any (optimal) integer point in \( Y^{IP}(\omega, x) \). To accomplish this, we will evaluate each of the components of \( y(\omega) \) and add an integer point \( \hat{y}(\omega) \) to the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) if it is the closest integer point to \( y^{LP}(\omega) \) for that component, or if all other integer points between \( \hat{y}(\omega) \) and the boundary of \( Y^{LP}(\omega, x) \) are already in the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \). There should be at least one integer point for every component \( i \in I \). Also, for any valid inequalities which are facets there will be at least \( n_2 \) affinely independent points in the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \).

### 3.2 Integer set generation algorithm

Based on Theorem 3.3, there exists \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) such that \( |\hat{Y}^{IP}(\omega, x, y^{LP})| = n_2 \) since subproblem (3) is full dimensional. However, getting the smallest set is not trivial unless we have an oracle providing an ideal interior point \( \hat{y}(\omega) \in Y^{IP}(\omega, x) \) on which the smallest set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) can be constructed. We devise an algorithm to obtain the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) based on the theory developed in the previous subsection. In the algorithm, we start with an initial point \( \hat{y}(\omega) \in \hat{Y}^{IP}(\omega, x, y^{LP}) \), where \( \hat{y}(\omega) \) is constructed based on \( y^{LP}(\omega) \). We then check whether the set of points in \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) are sufficient for generating a valid inequality, if not the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) is expanded sequentially by adding integer points from the set \( Y^{IP}(\omega, x) \). We refer to this algorithm for obtaining the set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) as Integer Set Generation (ISG). The ISG algorithm can be stated as follows:
Algorithm 1 Integer Set Generation (ISG) algorithm

Step [1] Initialize: Set $\hat{y}_i(\omega) = \lfloor y_{iLP}(\omega) \rfloor$, for all $i \in I$. Let $K' \subseteq K$ be the subset of indices for the binding constraints at $y_{iLP}(\omega)$.

Step [2] Evaluate bounds:

for all $i \in I$, $j \in I \setminus i$, $k \in K'$ do

$\alpha \leftarrow 0$;

repeat

$\alpha \leftarrow 0$;

Calculate $d_j(\omega, x, \hat{y}(\omega))$ using (8);

[2.a] Evaluate $d_j(\omega, x, \hat{y}(\omega))$:

if $(d_j(\omega, x, \hat{y}(\omega)) < 1)$ then

$\hat{y}_i(\omega) \leftarrow \hat{y}_i(\omega) - 1$; $\alpha \leftarrow 1$;

end if

[2.b] Check for integer points:

$b \leftarrow 0$;

while $\hat{y}_i(\omega) - b > 0 \& \alpha = 0$ do

Use $(\hat{y}_i(\omega) - b)$ to calculate $d_j(\omega, x, \hat{y}(\omega))$;

Use $(\hat{y}_i(\omega) - 1 - b)$ to calculate $d_j(\omega, x, \hat{y}(\omega))$;

if $(|\hat{d}_j(\omega, x, \hat{y}(\omega))| \leq u_i)$ then

$\hat{y}_i(\omega) \leftarrow \hat{y}_i(\omega) - b$; $\alpha \leftarrow 1$;

end if

$b \leftarrow b + 1$;

end while

$z \leftarrow z + 1$;

until $\alpha = 0$;

end for

Step [3] Use the computed lower bound: set $\hat{y}(\omega)$ as lower bound for the variables in problem (3).

In Algorithm 1, during each iteration, we evaluate two components of $y(\omega)$ along $i$ and $j$, and the lower bounds for the components are decreased based on the distance of the components from the binding constraints. Each pair of components, $\hat{y}_i(\omega)$ and $\hat{y}_j(\omega)$, are assigned to an integer point in two-dimensional (2D) space. The pair of components $(\hat{y}_i(\omega), \hat{y}_j(\omega))$ are evaluated in the 2D space with the criterion that they provide at least one integer point for the generation of the valid inequality in $i$-th direction. We also make sure that any integer point in $\text{conv}(\hat{Y}_{i}(\omega), x, y_{iLP}(\omega))$ is not removed so that the generated valid inequality does not cut off an optimal solution.

In Step [1], we initialize the parameter $\hat{y}(\omega)$ using the LP-relaxation solution to subproblem (3) given as $y_{iLP}(\omega)$ for $i \in I$. The parameter $\hat{y}_i(\omega)$ will be used as lower bound for the $i$-th component in subproblem (3). Ideally, we like to have a value as high as possible for $\hat{y}_i(\omega)$ without cutting off any integer solution to the original problem so the resulting $\hat{Y}(\omega, x, y_{LP})$ set is small. Initially $\hat{y}_i(\omega)$ is set to $\lfloor y_{iLP}(\omega) \rfloor$. In Step [2], for any $i, j \in I$, such that $i \neq j$, we calculate the distance $d_j(\omega, x, \hat{y}(\omega))$ in 2D space using $y_{jLP}(\omega)$ and equation (8) for the binding constraints of $y_{iLP}(\omega)$ for $k \in K'$. We then evaluate $d_j(\omega, x, \hat{y}(\omega))$ in Step [2.a]. The parameter $d_j(\omega, x, \hat{y}(\omega))$ is the distance measured from the other component’s axis to the binding constraint. If $d_j(\omega, x, \hat{y}(\omega)) < 1$, then it indicates the absence of an integer point along $j$-th axis.
then $\hat{y}_i(\omega)$ is decreased by one, and $\hat{y}(\omega)$ will be evaluated again for the binding constraint $k$ for the expanded set $\hat{Y}^{IP}(\omega, x, y^{LP})$. Hence, we start with a smallest set of integers based on $\lfloor y^{LP}_i(\omega) \rfloor$, and as the algorithm progresses, the set $\hat{Y}^{IP}(\omega, x, y^{LP})$ is expanded by decreasing the value of $\hat{y}(\omega)$.

In Step [2.b], we check if $d_j(\omega, x, \hat{y}(\omega)) \geq 1$. If there are any integer points along the component $i$, then $\hat{y}_i(\omega)$ is reduced to accommodate additional integer points for the set $\hat{Y}^{IP}(\omega, x, y^{LP})$. We make sure that the reduced set $\text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP}))$ is sufficient to get the required valid inequality to remove $y^{LP}_i(\omega)$ in the solution space. The complexity of the algorithm is $O(n^3m)$, where $n$ is the number of variables, and $m$ is the number of constraints. It should be noted that (A5) is a mild assumption, since the reduction of integer set along the axes with zero for $y^{LP}_i(\omega)$ is not required.

A representative illustration of ISG algorithm is presented in Fig. 2. The initial solution $\hat{y}(\omega) = (3, 2)$ is iterated couple of times in $\hat{y}_1(\omega)$ direction to obtain $(1, 2)$. Eventually, $\hat{y}_1(\omega)$ and $\hat{y}_2(\omega)$ get one and two as lower bounds, respectively, as a result of ISG algorithm. The 2D construction from the entire polytope is similar to variable elimination method described in [16]. Additionally, the above algebraic method is devised for the generation of Fenchel cuts where the coefficients of valid inequality are nonnegative. For other types of valid inequalities, appropriate transformations of the above algebraic method are required.

### 3.3 Numerical examples

Let us now illustrate the ISG algorithm using numerical examples. In Example 1, we use an IP subproblem with two decision variables and one constraint to illustrate the generation of a reduced integer set using the ISG algorithm. Example 2 demonstrates the algorithm for a subproblem with two decision variables and two constraints. In both examples, Step [2.b] of the ISG algorithm is not required, therefore we use Example 3 to demonstrate the significance of this step. Also, in the context of SIP2, for a given $(x, \omega)$ in a subproblem, the right hand side deterministic vector is equal to $h(\omega) - T(\omega)x$.

![Illustration of ISG algorithm](image-url)
**Example 1** Consider the following IP subproblem for a scenario \( \omega \):

\[
\text{IP1: Max } y_1(\omega) + y_2(\omega) \\
\text{s.t. } 0.4y_1(\omega) + y_2(\omega) \leq 3.4 \\
0 \leq y_1(\omega) \leq 3 \\
0 \leq y_2(\omega) \leq 3 \\
y_1(\omega), y_2(\omega) \in \mathbb{Z}_+.
\]

The LP-relaxation to problem (9) has the optimal solution \((3, 2.2)\) (see Fig. 3a). Thus, a valid inequality has to cut off this fractional solution without cutting off any integer points.

The steps of Algorithm 1 are as follows:

1. Step [1] Initialize: \( \hat{y}(\omega) = ([3], [2.2]) = (3, 2) \). Therefore, \( \hat{y}_1(\omega) = 3, \hat{y}_2(\omega) = 2 \), based on \( y_{1LP}(\omega) = 3 \) and \( y_{2LP}(\omega) = 2.2 \).
2. Step [2] Evaluate Bounds:

   - \( z=0, i=1, j=2, k=1 \): Calculate \( d_2(\omega, x, (3, 2)) \):
     
     \[
     r_1(\omega, x, (3, 2)) = (3.4 - 0.4(3))/1 = 2.2 \\
     \hat{d}_2(\omega, x, (3, 2)) = \min \{2.2 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega)\} = 0.2. \\
     d_2(\omega, x, (3, 2)) < 1 \Rightarrow \hat{y}_1(\omega) = \hat{y}_1(\omega) - 1 = 3 - 1 = 2, \text{ and } \alpha = 1.
     \]

   - \( z=1, i=1 \): Evaluate \( d_2(\omega, x, (2, 2)) \):
     
     \[
     r_1(\omega, x, (2, 2)) = (3.4 - 0.4(2))/1 = 2.6, \\
     \hat{d}_2(\omega, x, (2, 2)) = \min \{2.6/1 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega)\} = 0.6. \\
     d_2(\omega, x, (2, 2)) < 1 \Rightarrow \hat{y}_1(\omega) = \hat{y}_1(\omega) - 1 = 2 - 1 = 1, \text{ and } \alpha = 1.
     \]

   - \( z=2, i=1 \): Evaluate \( d_2(\omega, x, (1, 2)) \):
     
     \[
     r_1(\omega, x, (1, 2)) = (3.4 - 0.4(1))/1 = 3, \\
     \hat{d}_2(\omega, x, (1, 2)) = \min \{3 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega)\} = 1. \\
     Since, \( d_2(\omega, x, (1, 2)) \geq 1 \) and \( \alpha=0 \), then evaluate the next component \( \hat{y}_2(\omega) \):
     
     \[
     \hat{y}_2(\omega) = 2, \\
     r_1(\omega, x, (1, 2)) = (3.4 - 2)/0.4 = 3.5, \\
     d_1(\omega, x, (1, 2)) = \min \{3.5 - \hat{y}_1(\omega), 3 - \hat{y}_1(\omega)\} = \min \{3.5 - 1, 3 - 1\} = 2.
     \]

![Fig. 3](example.png)

**Fig. 3** Example 1 illustration: a reduced integer set and b cut generated based on the reduced set (dashed line)
• \( z=0, i=2 \): Since \( d_1(\omega, x, (1, 2)) \geq 1 \), we do not make any changes to \( \hat{y}_2(\omega) \).

3. Step [3] New Lower Bounds: Since \((\hat{y}_1(\omega), \hat{y}_2(\omega)) = (1, 2), y_1(\omega) \geq 1 \) and \( y_2(\omega) \geq 2 \).

Let \( \hat{y}(\omega) \) be an integer point assigned in the above iterations. The value \( d_2(\omega, x, \hat{y}(\omega)) = 0.2 \) represents the distance between the point \( \hat{y}(\omega) = (3, 2) \) and the point \( y_{LP}(\omega) = (3, 2.2) \) for a binding constraint in the \( \hat{y}_2(\omega) \) direction. Since there is no integer point in the direction, the algorithm iterates to assign the point \( \hat{y}(\omega) = (2, 2) \), where the distance \( d_2(\omega, x, (2, 2)) = \min \{2.6 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega)\} = 0.6 \).

The value 0.6 is the distance between the point \( \hat{y}(\omega) = (2, 2) \) and the binding constraint in \( \hat{y}_2(\omega) \) (vertical axis) direction. Since \( d_2(\omega, x, (2, 2)) < 1 \), the algorithm is continued to next iteration. Then the value for \( d_2(\omega, x, (2, 2)) \) is \( d_2(\omega, x, (2, 2)) = \min \{3 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega)\} = 1 \), which is the distance between the point \( \hat{y}(\omega) = (1, 2) \) and the binding constraint in \( \hat{y}_2(\omega) \) direction.

The feasible set based on the new point \((1, 2)\) as the origin is depicted in Fig. 3a. The reduced feasible set is now used for the generation of a cut, which is depicted in Fig. 3b.

**Example 2** Consider the following IP subproblem for a scenario \( \omega \):

\[
\text{IP2: Max } y_1(\omega) + y_2(\omega) \\
\text{s.t. } 0.4y_1(\omega) + y_2(\omega) \leq 3.4 \\
y_1(\omega) + 0.4y_2(\omega) \leq 3.4 \\
0 \leq y_1(\omega) \leq 3 \\
0 \leq y_2(\omega) \leq 3 \\
y_1(\omega), y_2(\omega) \in \mathbb{Z}_+. 
\]

The LP-relaxation to the problem (10) has the optimal solution \((2.42, 2.42)\) (see Fig. 4a). Thus, a valid inequality has to cut off the fractional solution. The steps of Algorithm 1 are as follows:

![Fig. 4 Example 2 illustration: a reduced integer set and b cut generated based on the reduced set (dashed line)](image-url)
1. Step [1] Initialize: \( \hat{y}(\omega) = (2.42, 2.42) = (2, 2) \). Therefore, \( \hat{y}_1(\omega) = 2 \), \( \hat{y}_2(\omega) = 2 \), based on \( y_1^{LP}(\omega) = 2.42 \) and \( y_2^{LP}(\omega) = 2.42 \).

2. Step [2] Evaluate Bounds:

- \( z=0, i=1, j=2, k=1 \): Calculate \( d_2(\omega, x, (2, 2)) \):

  \[ r_1^2(\omega, x, (2, 2)) = (3.4 - 0.4(2))/1 = 2.6 \]

  \[ d_2(\omega, x, (2, 2)) = \min \{ 2.6 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega) \} = 0.6. \]

- \( z=1, i=1, \) Evaluate \( d_2(\omega, x, (1, 2)) \):

  \[ r_1^2(\omega, x, (1, 2)) = (3.4 - 0.4(1))/1 = 3, \]

  which leads to \( d_2(\omega, x, (1, 2)) = \min \{ 3 - \hat{y}_1(\omega), 3 - \hat{y}_1(\omega) \} = 1. \)

  \[ k=1: d_2(\omega, x, (1, 2)) \geq 1, \] then evaluate the next constraint.

  \[ i=1, j=2, k=2, \]

  \[ r_2^2(\omega, x, (1, 2)) = (3.4 - 1)/0.4 = 2.4/0.4 = 6. \]

  \[ d_2(\omega, x, (1, 2)) = \min \{ 6 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega) \} = 1. \]

- \( z=0, i=1, k=2 \): \( d_2(\omega, x, (1, 2)) \geq 1, \) and all the constraints are evaluated, so we move to the next component \( \hat{y}_2(\omega) \).

- \( z=0, i=2, j=1, k=1 \): Similarly, for \( \hat{y}_2(\omega) = 2, r_1^2(\omega, x, (1, 2)) = (3.4 - 2)/0.4 = 3.5. \)

- \( d_1(\omega, x, (1, 2)) = \min \{ 3.5 - \hat{y}_1(\omega), 3 - \hat{y}_1(\omega) \} = 2. \]

  \[ d_1(\omega, x, (1, 2)) \geq 1, \] then evaluate the next constraint.

- \( z=0, i=2, j=1, k=2 \): \( r_2^2(\omega, x, (1, 2)) = (3.4 - 0.4(2))/1 = 2.6. \)

  \[ d_1(\omega, x, (1, 2)) = \min \{ 2.6 - \hat{y}_1(\omega), 3 - \hat{y}_1(\omega) \} = 1.6. \]

  Since \( d_1(\omega, x, (1, 2)) \geq 1, \) we do not make any changes to \( \hat{y}_2(\omega) \).

3. Step [3] New Lower Bounds: Since \( (\hat{y}_1(\omega), \hat{y}_2(\omega)) = (1, 2, 1), \) \( y_1(\omega) \geq 1 \) and \( y_2(\omega) \geq 2 \).

The value \( d_2(\omega, x, (2, 2)) = 0.6 \) represents the distance between the point \( \hat{y}(\omega) = (2, 2) \) and the point \( y^{LP}(\omega) = (2, 2.6) \) for the binding constraint (first constraint in (10)) in \( y_2(\omega) \) component’s direction. Since there is no integer point in the direction, the algorithm iterates to assign the point \( \hat{y}(\omega) = (1, 2), \) where the distance \( d_2(\omega, x, (1, 2)) = \min \{ 3 - \hat{y}_2(\omega), 3 - \hat{y}_2(\omega) \} = 1. \) This value is the distance between the point \( \hat{y}(\omega) = (1, 2) \) and the binding constraint (first constraint in (10)) in \( \hat{y}_2(\omega) \) direction. Since \( d_2(\omega, x, (1, 2)) \geq 1, \) the algorithm considers the next constraint. Similarly, the other index \( \hat{y}_2(\omega) \) is evaluated in \( \hat{y}_1(\omega) \)'s direction for both the first and second constraints in (10).

The feasible set based on the new origin \((1, 2)\) is depicted in Fig. 4a. The feasible set is used for the generation of valid inequalities, and the generated cut is shown in Fig. 4b. After performing the ISG algorithm, the new origin is \((1, 2)\).

**Example 3** Consider another IP subproblem given as follows for a scenario \( \omega \):

**IP3:**

\[
\begin{align*}
\text{Max} \quad & 1.2y_1(\omega) + 3.4y_2(\omega) \\
\text{s.t.} \quad & 6y_1(\omega) + 5y_2(\omega) \leq 37.4 \\
& 0 \leq y_1(\omega) \leq 5 \\
& 0 \leq y_2(\omega) \leq 5 \\
& y_1(\omega), y_2(\omega) \in \mathbb{Z}_+. 
\end{align*}
\]
The LP-relaxation of problem (11) gives the solution (5, 1.48) (Fig. 5a). Using Step [2.a] of the ISG algorithm, the new origin is shifted to (4, 1). However, based on $\bar{y}^{LP}(\omega, x, y^{LP})$, the generated inequality removes an integer point (2, 5) from the solution space. Hence, we use Step [2.b] to prevent the possibility of removing any integer points from the solution space based on the current reference obtained from Step [2.b]. Thus Step [2.b] gives the new reference point (2, 1) based on the possible integer points in $y^{LP}(\omega, x)$. The reduced solution space based on the new origin (4, 1) without Step [2.b] is shown in Fig. 5(a), while the reduced solution space based on Step [2.b] for the generation of a valid inequality to remove $y^{LP}(\omega)$ is shown in Fig. 5b. Notice that the inequality generated is valid since it does not cut off any integer points.

The LP-relaxation to the problem (11) has the optimal solution (5, 1.48) (see Fig. 5a). Using $y^{LP}(\omega)$, the generated inequality removes an integer point (2, 5) from the solution space. Hence, we use Step [2.b] to prevent the possibility of removing any integer points from the solution space based on the current reference obtained from Step [2.b]. Thus Step [2.b] gives the new reference point (2, 1) based on the possible integer points in $y^{LP}(\omega, x)$. The reduced solution space based on the new origin (4, 1) without Step [2.b] is shown in Fig. 5(a), while the reduced solution space based on Step [2.b] for the generation of a valid inequality to remove $y^{LP}(\omega)$ is shown in Fig. 5b. Notice that the inequality generated is valid since it does not cut off any integer points.

The LP-relaxation to the problem (11) has the optimal solution (5, 1.48) (see Fig. 5a). Thus, a valid inequality has to cut off the fractional solution. The steps of Algorithm 1 are as follows:

1. Step [1] Initialize: $\hat{y}(\omega) = ([5], [1.48]) = (5, 1)$. Therefore, $\hat{y}_1(\omega) = 5$, $\hat{y}_2(\omega) = 1$, based on $y_1^{LP}(\omega) = 5$ and $y_2^{LP}(\omega) = 1.48$.

2. Step [2a] Evaluate Bounds:
   - $z=0$, $i=1$, $j=2$, $k=1$: Calculate $d_2(\omega, x, (5, 1))$: $r_1(\omega, x, (5, 1)) = (37.4 - 6(5))/5 = 1.48$, $\bar{d}_2(\omega, x, (5, 1)) = \min \{ 1.48 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 0.48$. $d_2(\omega, x, (5, 1)) < 1 \Rightarrow \hat{y}_1(\omega) = \hat{y}_1(\omega) - 1 = 5 - 1 = 4$, update $\hat{y}(\omega) \leftarrow (4, 1)$, and $\alpha = 1$.
   - $z=1$, $i=1$, Evaluate $d_2(\omega, x, (4, 1))$: $r_1 = (37.4 - 6(4))/5 = 2.68$, which leads to $d_2(\omega, x, (4, 1)) = \min \{ 2.68 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 1.68$. $d_2(\omega, x, (4, 1)) \geq 1$.
   - Step [2b]: $z=1$, $i=1$, Evaluate $d_2(\omega, x, (3, 1))$: $r_1 = (37.4 - 6(3))/5 = 3.88$, which leads to $d_2(\omega, x, (3, 1)) = \min \{ 3.88 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 2.88$. Since $[\bar{d}_2(\omega, x, (3, 1))] = [d_2(\omega, x, (4, 1))] \geq 1 \Rightarrow \hat{y}_1(\omega) = \hat{y}_1(\omega) - 1 = 4 - 1 = 3$, update $\hat{y}(\omega) \leftarrow (3, 1)$, and $\alpha = 1$.

Fig. 5 Example 3 illustration: a incorrect reduced integer set and b reduced integer set
• \( z=2, i=1 \), Evaluate \( d_2(\omega, x, (3, 1)) = r_2 = (37.4 - 6(3))/5 = 3.88 \), which leads to \( d_2(\omega, x, (3, 1)) = \min \{ 3.88 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 2.88 \). \( d_2(\omega, x, (3, 1)) \geq 1 \).

• Step [2b]: \( z=2, i=1 \), Evaluate \( d_2(\omega, x, (2, 1)) = r_2 = (37.4 - 6(2))/5 = 5.08 \), which leads to \( d_2(\omega, x, (2, 1)) = \min \{ 5.08 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 4 \). Since \( [d_2(\omega, x, (2, 1))] - [d_2(\omega, x, (3, 1))] \geq 1 \) \( \Rightarrow \hat{y}_1(\omega) = \hat{y}_1(\omega) - 1 = 3 - 1 = 2 \), update \( \hat{y}(\omega) \leftarrow (2, 1) \), and \( \alpha = 1 \).

• \( z=3, i=1 \), Evaluate \( d_2(\omega, x, (2, 1)) \): \( r_2 = (37.4 - 6(2))/5 = 5.08 \), which leads to \( d_2(\omega, x, (2, 1)) = \min \{ 5.08 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 4 \). \( d_2(\omega, x, (2, 1)) \geq 1 \).

• Step [2b]: \( z=3, i=1 \), Evaluate \( d_2(\omega, x, (1, 1)) = r_2 = (37.4 - 6(1))/5 = 6.28 \), which leads to \( d_2(\omega, x, (1, 1)) = \min \{ 6.28 - \hat{y}_2(\omega), 5 - \hat{y}_2(\omega) \} = 4 \). Since \( [d_2(\omega, x, (2, 1))] - [d_2(\omega, x, (3, 1))] \leq 0 \). No changes to \( \hat{y}_1(\omega) \). We move to the next component \( \hat{y}_2(\omega) \).

• \( z=0, i=2, j=1, k=1 \). Similarly, for \( \hat{y}_2(\omega) = 1 \), \( r_1(\omega, x, (2, 1)) = (37.4 - 5(1))/6 = 5.4 \)
  \( d_1(\omega, x, (2, 1)) = \min \{ 5.4 - \hat{y}_1(\omega), 5 - \hat{y}_1(\omega) \} = 3 \). \( d_1(\omega, x, (2, 1)) \geq 1 \). We do not make any changes to \( \hat{y}_2(\omega) \).

3. Step [3] New Lower Bounds: Since \( (\hat{y}_1(\omega), \hat{y}_2(\omega)) = (2, 1) \), \( y_1(\omega) \geq 2 \) and \( y_2(\omega) \geq 1 \).

The examples demonstrated the steps of the ISG procedure. Eventually, the procedure adjusts the lower bounds for the variable \( \gamma(\omega) \) to reduce the subproblem’s solution space. The newly computed lower bounds will be used for the generation of valid inequalities. The objective of ISG algorithm is to obtain a relatively smaller set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \subseteq Y^{IP}(\omega, x) \) so that \( \text{conv}(\hat{Y}^{IP}(\omega, x, y^{LP})) \subseteq \text{conv}(Y^{IP}(\omega, x)) \). A reduced set \( \hat{Y}^{IP}(\omega, x, y^{LP}) \) is expected to provide better runtime for solving the instances of SMIP.

### 4 Fenchel decomposition algorithm

To corroborate the efficiency of ISG algorithm within a decomposition procedure for solving SMIP instances, we apply it to stage-wise Fenchel decomposition (SFD) for solving SMIPs [28]. SFD adopts the Benders’ decomposition setting with \( x \) as the first-stage decision variable in the master problem, and \( \gamma(\omega) \) as the second-stage decision variable in the subproblem. In SIP2, instead of working with the IP subproblem directly, SFD seeks to find the optimal solution via a cutting plane approach on a partial LP-relaxation of SIP2 where only the subproblems are relaxed. Fenchel decomposition (FD) cuts are sequentially generated to recover (at least partially) the convex hull of integer points for each scenario subproblem feasible set. If a subproblem LP has a non-integer solution, an FD is generated and added to cut off the fractional solution. FD cuts are capable of recovering faces of the convex hull of integer programs. The goal is to construct the convex hull of integer points in the neighborhood of the optimal solution so that by solving subproblem LPs with sufficient FD cuts added, we can find the optimal solution without having to resort to a branch-and-bound scheme to guarantee optimality.
4.1 Algorithm

At a given iteration $\ell$ of the SFD cutting plane algorithm, the master problem takes the following form:

$$z^\ell = \text{Max } \mathbf{c}^\top \mathbf{x} + \theta$$

s.t. $A\mathbf{x} \leq \mathbf{b}$

$$(\eta^t)^\top \mathbf{x} + \theta \leq \gamma^t$, $t \in \{1, ..., \ell\}$$

$x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_1-n'_1}$.

In the master problem (12), $\theta \in \mathbb{R}$ is the optimality cut decision variable, $\eta^t \in \mathbb{R}^{n_1}$ is the optimality cut coefficient vector generated at iteration $t$, and $\gamma^t \in \mathbb{R}$ is the corresponding right hand side. Constraints with $\theta$ variable are the optimality cuts, which are computed based on the optimal dual solutions of all the LP-relaxation subproblems. Optimality cuts approximate the expected recourse function. For a first-stage solution $x^\ell$ from the master problem (12), the subproblem $\text{SP}(\omega)$ for each scenario $\omega \in \Omega$ is given as follows:

$$\Phi^{\ell}_{LP}(q(\omega), \tau(\omega, x), \omega) = \text{Max } q(\omega)^\top y(\omega)$$

s.t. $Wy(\omega) \leq \tau(\omega, x)$

$$\alpha^t(\omega)^\top x + \beta^t(\omega)^\top y(\omega) \leq g(\omega, \alpha^t(\omega), \beta^t(\omega)),$$

$t \in \Theta(\omega)$

$0 \leq y(\omega) \leq u.$

Constraints with $\alpha$ and $\beta$ coefficients for $x$ and $y(\omega)$ variables, respectively, are the Fenchel cuts, and $\Theta(\omega)$ is the index set for algorithm iterations at which a Fenchel cut is generated for each $\omega \in \Omega$. The coefficients $\alpha^t(\omega)$ and $\beta^t(\omega)$ for the $x$ and $y(\omega)$ variables, respectively, and the right hand side $g(\omega, \alpha^t(\omega), \beta^t(\omega))$ are obtained as explained in the next section. Next, we first describe how these cuts are generated.

The SFD algorithm starts by initializing data in Step [1] and getting an initial solution by solving the LP-relaxation of SIP2 in Step [2]. If the initial solution satisfies the integrality restrictions for all subproblems in Step [3], i.e., $x^0 \in X$ and $y^{LP}(\omega) \in Y^{LP}(\omega, x)$, $\forall \omega \in \Omega$, then the solution is declared $\epsilon$-optimal, and the algorithm terminates. Otherwise, the algorithm continues by calculating and storing the optimality cut coefficients for all subproblems with an integer solution in Step [4].
Algorithm 2 Stage-wise Fenchel Decomposition (SFD) algorithm

1. Step [1] Initialization: set $\ell \leftarrow 0$, $\epsilon > 0$, $LB \leftarrow -\infty$ and $UB \leftarrow \infty$.
2. Step [2] Get initial solution: Solve problem (12-13) using the L-shaped algorithm to get solution $(x^0, y^{L,P}(\omega), \phi^0 = \sum_{\omega \in \Omega} p_{\omega} \Phi_{L,P}(g(\omega), \tau(\omega, x, \omega))$, and dual solutions $\pi^0(\omega)$ for the constraints $W g(\omega) \leq \tau(\omega, x) \varepsilon$ for each $\omega \in \Omega$.
3. Step [3] Check solution integrality:
   a. if $y^{L,P}(\omega) \in \mathbb{Z}_+$, $\forall \omega \in \Omega$ then
      Report $(x^0, y^{L,P}(\omega))$ as optimal.
   b. Stop.
4. Step [4] Calculate and store optimality cuts coefficients for scenarios with integer solution
   for $\omega \in \Omega$ do
   10. if $y^{L,P}(\omega) \in Y^{IP}(\omega, x)$ then
       Calculate and store optimality cut coefficients $\eta^\ell \leftarrow \eta^\ell + p_{\omega}(\pi^\ell(\omega))^T T(\omega)$ and $\gamma^\ell \leftarrow \gamma^\ell + p_{\omega}(\pi^\ell(\omega))^T h(\omega)$.
      end if
   11. end for
5. Step [5] Fenchel cuts and optimality cuts generation:
   for $\omega \in \Omega$ do
   16. if $y^{L,P}(\omega) \notin Y^{IP}(\omega, x)$ then
      Compute scenario Fenchel cut coefficients: Run ISG to get $\hat{Y}^{IP}(\omega, x, y^{L,P})$ and use Fenchel cut generation procedure based on $\hat{Y}^{IP}(\omega, x, y^{L,P})$ to get $\alpha^\ell(\omega), \beta^\ell(\omega)$ and $g(\omega, \alpha^\ell(\omega), \beta^\ell(\omega))$. Add the cut $\alpha^\ell(\omega)^T x + \beta^\ell(\omega)^T g(\omega) \leq g(\omega, \alpha^\ell(\omega), \beta^\ell(\omega))$ to subproblem (13). Solve the updated subproblem $SP(\omega)$ and get updated subproblem dual solution $\pi^\ell(\omega)$.
   17. Update optimality cut coefficients $\eta^\ell \leftarrow \eta^\ell + p_{\omega}(\pi^\ell(\omega))^T T(\omega)$ and $\gamma^\ell \leftarrow \gamma^\ell + p_{\omega}(\pi^\ell(\omega))^T h(\omega)$.
      end if
   18. end for
6. Step [6] Add optimality cut $(\eta^\ell)^T x + \theta \leq \gamma^\ell$ to master problem (12), and update iteration index: $\ell \leftarrow \ell + 1$.
7. Step [7] Solve master problem (12) to get a new first-stage solution $x^\ell$ and objective value $z^\ell$.
8. Step [8] Update bounds:
   26. Set $UB \leftarrow z^\ell$
   27. Update incumbent solution: $x^* \leftarrow x^\ell$
   28. Update lower bound: $LB \leftarrow \max\{LB, c^T x^\ell + \sum_{\omega \in \Omega} p_{\omega} \Phi_{L,P}(g(\omega), \tau(\omega, x, \omega))\}$
9. Step [9] $\epsilon$-optimality check:
   30. if $|UB - LB| > \epsilon |LB|$ then
      Solve the updated subproblems $SP(\omega), \forall \omega \in \Omega$.
   31. Go to Step [4].
   32. end if
10. Step [10] Declare $x^*$ as $\epsilon$-optimal.

For subproblems with solutions that does not satisfy the integrality requirements, corresponding Fenchel cut coefficients $\alpha^\ell(\omega)$ and $\beta^\ell(\omega)$, and the right hand side $g(\omega, \alpha^\ell(\omega), \beta^\ell(\omega))$ are computed for the iteration $\ell$ in Step [5]. A Fenchel cut is added to subproblem $SP(\omega)$. Next, the dual solution obtained by solving the subproblem is used to generate the optimality cut coefficients. Once all subproblems have been solved at a given iteration, the optimality cut is added to the master problem in Step [6]. The
iteration counter $\ell'$ is incremented by one, and the master problem is solved again in Step [7] to get an updated first-stage solution and objective value.

The upper bound $UB$, the incumbent solution $x^*$ and the lower bound $LB$ are updated in Step [8]. The gap between the lower bound $LB$ and the upper bound $UB$ is verified in Step [9]. If this gap is small enough, then the incumbent solution is declared $\epsilon$-optimal in Step [10], and then the algorithm terminates. Otherwise, all the subproblems are solved again, and algorithm continues from Step [4]. The algorithm is continued until the termination condition is satisfied.

### 4.2 Fenchel cut generation with reduced integer set

For the sake of completeness, we next present the Fenchel Cut Generation (FCG) procedure which is based on [28]. The procedure uses a master problem to construct a linear approximation of the subproblem space while the subproblem returns feasible integer points from $\hat{Y}_{IP}(\omega, x, y_{LP})$. For a non-integer solution $y_{LP}(\omega')$ during the iteration $\nu$ in Algorithm 2, the master problem for FCG procedure for iteration $\nu$ is given as:

$$\delta_{\nu} = \max_{(\alpha(\omega), \beta(\omega)) \in \Pi^{\alpha, \beta}} \theta$$

s.t. 

$$-\theta + (x' - x^T \alpha(\omega) + (y_{LP}(\omega') - y_{LP}(\omega)) \beta(\omega) \geq 0, \forall \ t = 1, \ldots, \nu,$$

(14)

$\alpha(\omega)$ and $\beta(\omega)$ are chosen from some convex set $\Pi^{\alpha, \beta}$. A linearly constrained domain for $\Pi^{\alpha, \beta}$ such as the $L^1$ unit sphere, $\Pi^{\alpha, \beta} = \{(\alpha(\omega), \beta(\omega)) \in \mathbb{R}_{+}^{n_1 + n_2} | 0 \leq \alpha(\omega) \leq 1, 0 \leq \beta(\omega) \leq 1\}$ provides a better choice in terms of solution time. Based on $\Pi^{\alpha, \beta}$, a free variable $\theta$ in (14) enables to compute the maximum distance between the non-integer point $(x', y_{LP}(\omega'))$ and the Fenchel cut. The integer point for the iteration $t$ given as $(x', y_{IP}(\omega'))$ is obtained via solving the following subproblem at iteration $t$:

$$g(\omega, \alpha'(\omega), \beta'(\omega)) = \max_{x, y(\omega) \in \hat{Y}_{IP}(\omega, x, y_{LP})} \left\{ \alpha'(\omega)^T x + \beta'(\omega)^T y(\omega) \right\}.$$

(15)

Solving the problems (14) and (15) iteratively will give the optimal $\alpha(\omega)$, $\beta(\omega)$ and $g(\omega, \alpha(\omega), \beta(\omega))$. The algorithm is stated as follows:
In Step [1] we initialize the parameters for the algorithm. Components $\alpha^{0}(\omega), \beta^{0}(\omega)$ are arbitrarily initialized to a value within their bounds. Since an integer subproblem (3) has to be solved many times to generate Fenchel cuts, a linearly constrained domain for $\Pi^\alpha, \beta^\alpha$ such as the $L^1$ unit sphere norm can be used. For $L^1$ norm, each of the variable is restricted between 0 and 1. Step [2] uses $\alpha^0(\omega)$ and $\beta^0(\omega)$ as coefficients, then subproblem (15) is solved, and the corresponding objective value is stored. Due to ISG procedure, subproblem (15) can be evaluated using $\hat{Y}^{IP}(\omega, x, y^{LP})$ instead of $Y^{IP}(\omega, x)$. It should be noted that subproblem (15) is solved as an IP so the solutions $x^t$ and $y^t(\omega)$ are integral. The lower bound and incumbent solution are updated in Step [2]. Based on the solutions $x^t$ and $y^t(\omega)$ from subproblem (15), the cut is added to master problem (14). In Step [3], master problem (14) is solved and the upper bound is updated. Based on the lower and upper bounds, termination condition is checked. If the difference between the bounds are bigger than a user threshold value, the algorithm continues else terminates.

5 Computational study

To gain insights into using integer set reduction in a cutting plane algorithm, we performed a computational study based on an implementation of the SFD algorithm with the option of turning on ISG. The algorithm was implemented in Java programming language using CPLEX 12.8 Concert Library [15]. Computations were performed on an x64 computer with Intel®Xeon®Processor ES-2640 v3 (2.60GHz, 2...
processors) and 80GB RAM. CPLEX MIP and LP solvers were used to optimize the master program and subproblems. Three test sets (Set1, Set2, and Set3) were created for a two-stage multidimensional knapsack problem with special structure. The performance of SFD using ISG was benchmarked with a direct MIP solver applied to DEP given in (4) for the sets, and with our implementation of binary first-stage algorithm for SIP with pure binary first-stage decision variables [25] for Set1. We refer to this algorithm as ST-L2 algorithm. A DEP for each test instance was created and solved using the CPLEX MIP solver. Computational experiments were conducted on four types of test instances for each set (Set1, Set2, and Set3) from stochastic multidimensional knapsack problems. Next, we describe the formulation and test sets, and then report computational findings.

5.1 Test instances generation

To test our cutting plane decomposition algorithm, we created randomly generated test instances of a two-stage multidimensional knapsack problem. The first-stage problem is given as follows:

$$\text{Max } c^T x + Q_E(x)$$
$$\text{s.t. } \sum_{i \in I} x_i \leq b$$
$$x \in \{0, 1\}^{n_1}. \tag{16}$$

In problem (16), $c \in \mathbb{R}^{n_1}$ is the first-stage cost vector, and $b \in \mathbb{R}$ is the first-stage right hand side. The function $Q_E(x)$ is the expected recourse function and is given as

$$Q_E(x) = \mathbb{E}_{\tilde{\omega}} \Phi(x, \tilde{\omega}), \tag{17}$$

where $\tilde{\omega}$ is a multivariate random variable and $Q_E(.)$ denotes the mathematical expectation operator satisfying $\mathbb{E}_{\tilde{\omega}} [ | \Phi(x, \tilde{\omega}) | ] < \infty$. The underlying probability distribution of $\tilde{\omega}$ is discrete with a finite number of realizations (scenarios) with sample space $\Omega$, and corresponding probabilities $p_\omega$, $\omega \in \Omega$. Thus for a given scenario $\omega \in \Omega$, the recourse function $\Phi(x, \omega)$ is given by the following second-stage IP subproblem:

$$\Phi(x, \omega) = \text{Max } q(\omega)^T y(\omega)$$
$$\text{s.t. } W y(\omega) + Tx \leq 0$$
$$V y(\omega) \leq h(\omega)$$
$$0 \leq y(\omega) \leq u, \ y(\omega) \in \mathbb{Z}_+. \tag{18}$$

In formulation (18), $y(\omega)$ is the recourse decision vector, $q(\omega) \in \mathbb{R}^{n_2}$ is the recourse cost vector, $W \in \mathbb{R}^{m_2 \times n_2}$ is a fixed recourse parameter, and $T \in \mathbb{R}^{m_1 \times n_1}_+, V \in \mathbb{R}^{m_3 \times n_2}_+, h(\omega) \in \mathbb{R}^{m_3}$ is a right hand side parameter. The decision vector $y(\omega)$ is bounded above by vector $u$. Finally, $\mathbb{Z}_+$ is the set of nonnegative integers. Observe that formulation (16)-(18) has knapsack constraints in both the first and second-stages.
In a supply chain context, the model represents replenishment decisions subject to uncertainty in price and demand of the products. Replenishment decisions are made on periodic basis to determine the amount of raw materials or finished products to be procured from a downstream supply chain member. However, if there are large variations in transportation, inventory costs, or demand for the products then a stochastic model will be appropriate [46]. In the formulation (16)-(18), the first-stage decision vector $x$ specifies the selection of facilities, mode of transportation, products, and/or resources. For a realization $\omega$, the second-stage decision vector $y(\omega)$ is the amount of products produced or transported based on the strategic decision $x$ from the first-stage. Additionally, knapsack-type constraints are added to represent capacity or demand limitations in the second-stage subproblem.

Test instance data were randomly generated using the uniform distribution ($\mathcal{U}$) with different parameter values. The knapsack weights $W$ and $T$ were generated by sampling from $\mathcal{U}(2, 8)$. The first and second-stage objective function coefficients were generated by sampling from $\mathcal{U}(0, 1500)$ and $\mathcal{U}(10, 20)$, respectively. To generate tighter knapsack constraints, the right hand side value of each constraint was generated by finding the maximum knapsack weight ($W_{\text{max}}$) for the constraint, and then sampled from $\mathcal{U}(2 + (2W_{\text{max}} \cdot v_{ub}), 4W_{\text{max}} \cdot v_{ub})$, where $v_{ub}$ is the upper bound for the integer variables. We assumed that each scenario had equal probability of occurrence.

The problem characteristics for DEP are given in Table 1. The columns of the table are problem name, ‘Scens’ is the number of scenarios, ‘Bvars/Ivars’ is the number of binary or general integer variables in the first-stage, ‘Ivars’ is the number of integer variables in the second-stage subproblem, ‘Constr’ is the number of constraints, and ‘Nzeros’ is the number of non-zero elements for each of the problem instances. Test sets Set1 and Set3 have binary decision variables for the first-stage with $x \in \{0, 1\}^{10}$ and $x \in \{0, 1\}^{20}$, respectively. Set2 has general integer variables for the first-stage with $x \in \mathbb{Z}_+^{10}$.

| Instance | Scens | Bvars/Ivars | Ivars | Constr | Nzeros |
|----------|-------|-------------|-------|--------|--------|
| Set1     |       |             |       |        |        |
| S.10.20.50 | 50   | 10/0        | 1000  | 1010   | 12,510 |
| S.10.20.100 | 100  | 10/0        | 2000  | 2010   | 25,010 |
| S.10.20.150 | 150  | 10/0        | 3000  | 3010   | 37,510 |
| S.10.20.200 | 200  | 10/0        | 4000  | 4010   | 50,010 |
| Set2     |       |             |       |        |        |
| L.10.20.50 | 50   | 0/10        | 1000  | 1010   | 12,510 |
| L.10.20.100 | 100  | 0/10        | 2000  | 2010   | 25,010 |
| L.10.20.150 | 150  | 0/10        | 3000  | 3010   | 37,510 |
| L.10.20.200 | 200  | 0/10        | 4000  | 4010   | 50,010 |
| Set3     |       |             |       |        |        |
| L.20.30.500 | 500  | 20/0        | 15,000| 10,010 | 330,020|
| L.20.30.1000 | 1000 | 20/0        | 30,000| 20,010 | 660,020|
| L.20.30.1500 | 1500 | 20/0        | 45,000| 30,010 | 990,020|
| L.20.30.2000 | 2000 | 20/0        | 60,000| 40,010 | 1,320,020|
Sets 1 and 2 have 10 and 20 constraints in the first and second-stages, respectively, and the test instances were created with 50, 100, 150, and 200 scenarios for both the sets. Set 3 is indicative of large scale model with 500, 1000, 1500, and 2000 scenarios. Five randomly generated replications were created for each instance size to avoid pathological cases. The problem name has the form $k.m.n.S$, where $k$ stands for whether smaller(S) or larger(L) set, $m$ and $n$ are the number of first and second-stage decision variables, respectively, and $S$ is the number of scenarios.

### 5.2 Results

Detailed computational results for the test sets Set1, Set2, and Set3 are reported in (refer Appendix) Tables 2, 3 and 4, respectively. In Table 2, ‘SFD’ represents results using SFD algorithm, ‘SFD-R’ represents results using SFD algorithm with ISG algorithm, and ‘ST-L2’ represents the results using integer L-shaped algorithm [25]. ‘Run time’ represents the total run time in seconds for SFD, SFD-R, and ST-L2 algorithms to attain optimality for the instances, ‘# MIPs solved’ indicate the number of MIPs solved for generating Fenchel cuts, ‘Opt-Obj’ represents the optimal objective value attained using SFD, SFD-R, and ST-L2 algorithms, and finally, ‘%Gap’ represents CPLEX MIP gap % reported for the instances after one hour runtime. Optimal solutions were obtained using SFD, SFD-R, and ST-L2 algorithms for Set1 instances. However, optimality was not attained for any of the DEP instances, hence the final MIP gap after stipulated time of one hour is reported. In Tables 3 and 4, the data under SFD-R are obtained by using SFD-R algorithm, and the five columns are as follows: ‘MIPs’ is the number of MIPs solved using the ISG procedure for the generation of Fenchel cuts; ‘FCuts’ is the number of Fenchel cuts generated; and ‘%Gap’ is the percentage gap between the lower bound (LB) and the upper bound (UB) value after the stipulated runtime (7200s for Set2 and Set3). Similarly, ‘MIPs’ and ‘FCuts’ represent the number of MIPs solved and Fenchel cuts generated without using ISG procedure in the SFD algorithm. Finally, the last column ‘%Gap’ is the CPLEX MIP gap reported for DEP instances.

Figs. 6, 7 and 8 denote the performance of the algorithms for Set1, Set2, and Set3, respectively. The horizontal axis for Figs. 6, 7 and 8 represent the corresponding instances given in Table 1. For Set1, SFD, SFD-R, and ST-L2 obtained optimal solutions whereas CPLEX MIP solver applied to the DEP could not solve any of the instances to optimality. In Fig. 6, ‘Run time’ denotes the total runtime for SFD, SFD-R, and ST-L2 to obtain optimal solutions. The plot in Fig. 6a indicates that the total time taken by SFD and SFD-R for instances with less than 150 scenarios is about the same, however for instances with 150 or more scenarios, SFD-R shows a slight advantage over SFD. The results show that SFD-R has taken the least time to reach optimality compared to SFD (11% reduction) and ST-L2 (62% reduction). Fig. 6b shows that ST-L2 takes significantly higher runtime compared to SFD or SFD-R algorithms due to the weaker cuts in ST-L2 algorithm. ‘MIPs solved’ indicates the number of MIPs solved by SFD and SFD-R algorithms as part of FCG procedure as denoted by the problem (15). SFD-R uses lesser number of MIPs compared to SFD, hence solves the instances to optimality faster. This indicates that the cuts generated using SFD-R are deeper compared to SFD. Finally, ‘MIP Gap %’ in Fig. 6c is the MIP gap reported by CPLEX for DEP instances.
In Fig. 7a, ‘MIPs solved’ represents the number of MIPs solved as part of FCG procedure to generate Fenchel cuts using SFD-R and SFD algorithms. ‘Fenchel cuts’ in Fig. 7b denotes the number of Fenchel cuts generated using SFD-R and SFD algorithms, and finally, Gap(%) represents the final optimality gap using SFD-R and CPLEX MIP gap for corresponding DEP instances. SFD-R performs consistently well compared to SFD in terms of solving most number of MIPs and generating Fenchel cuts. On an average, SFD-R was able to solve 14%, 18%, 13%, and 14% more MIPs compared to SFD for the instances with 50, 100, 150, and 200 scenarios, respectively. With more MIPs solved, SFD-R was able to generate more Fenchel cuts compared to SFD and achieve better optimality gaps. On an average, SFD-R generated 14%, 17%, 14%, and 16% more Fenchel cuts compared to SFD for the instances with 50, 100, 150, and 200 scenarios, respectively. The performance of SFD-R was benchmarked by solving DEP instances using CPLEX, and the optimality gaps are reported in the ‘Optimality gap’ plot in Fig. 7c. Though SFD-R could
not solve any of the large instances to optimality, the algorithm performed better than a direct solver. It should be noted that CPLEX was able to utilize multi-threading well in our hardware, however the subproblems for SFD-R were solved serially. We would expect improvement in closing the optimality gaps if the subproblems of SFD-R algorithm are solved in parallel. Also, computational experiments for Set2 using ST-L2 were not performed because the algorithm is not applicable for $x \in \mathbb{Z}_+$.  

Fig. 8 shows characteristics for Set3 which are similar to Set2. With large number of scenarios in Set3, a direct solver could not find a feasible solution in more than 50% of instances. However, %Gaps for SFD-R were consistently below 1% for all the instances. On an average, SFD-R was able to solve 12%, 8%, 8%, and 12% more MIPs compared to SFD for the instances with 500, 1000, 1500, and 2000 scenarios, respectively, as shown in Fig. 8a. With more MIPs solved, SFD-R was able to generate more Fenchel cuts compared to SFD and achieve better optimality gaps. On an average, SFD-R generated 18%, 13%, 16%, and 18%
more Fenchel cuts compared to SFD for the instances with 500, 1000, 1500, and 2000 scenarios, respectively, as indicated in Fig. 8b.

6 Conclusion

This work introduces a new theory and algorithm for integer set reduction procedure for cutting plane methods for SMIP with general integer variables in the second-stage subproblem. Example illustrations of the new method in the context of generating Fenchel cutting planes are given. The method is then incorporated into the Fenchel decomposition algorithm for SMIP and a computational study is performed to assess the performance of the new approach. The results from the computational study show that incorporating integer set reduction in a cutting plane algorithm can result in better bounds and provides better performance than a direct solver applied to the deterministic equivalent problem. Also, more cuts are generated in a given time period when integer set reduction is used as opposed to when it is not used. Future work along this line of work includes extending the integer set reduction procedure to SMIP with arbitrary or general recourse matrices. Since the algorithm fits well for SIP2 with mean-risk measures like CVaR, excess probability, and expected excess, which have block angular structure conducive for stage-wise decomposition, a computational study for this class of SMIP in the future will be interesting. Another extension is to incorporate and evaluate the new procedure in other cutting plane methods for SMIP such as disjunctive decomposition and dual decomposition.

Appendix

See Tables 2, 3 and 4.
### Table 2  Computational results for Set1 instances (Run time: 3,600s)

| No. | Instance       | Run time(s) | # MIPs solved | Opt-Obj | CPLEX |
|-----|----------------|-------------|---------------|---------|-------|
|     |                | SFD | SFD-R | ST-L2 | SFD | SFD-R | %Gap |
| 1   | S.10.20.50a    | 404 | 426   | 1400  | 3795 | 3757  | 23.38 | 5.02 |
| 2   | S.10.20.50b    | 519 | 520   | 1967  | 4360 | 4219  | 42.78 | 5.08 |
| 3   | S.10.20.50c    | 482 | 450   | 1521  | 4143 | 3988  | 108.68| 5.46 |
| 4   | S.10.20.50d    | 439 | 435   | 1365  | 3771 | 3700  | 170.66| 6.24 |
| 5   | S.10.20.50e    | 479 | 428   | 833   | 4048 | 3778  | 281.98| 4.27 |
|     | Average        | 465 | 452   | 1417  | 4023 | 3888  | 5.21  |
| 6   | S.10.20.100a   | 1123| 1067  | 2527  | 8388 | 8197  | 77.13 | 4.89 |
| 7   | S.10.20.100b   | 905 | 885   | 2793  | 7665 | 7377  | 107.74| 4.55 |
| 8   | S.10.20.100c   | 1002| 945   | 2323  | 7785 | 7663  | 124.70| 5.21 |
| 9   | S.10.20.100d   | 934 | 928   | 2911  | 7653 | 7497  | 159.84| 5.29 |
| 10  | S.10.20.100e   | 1036| 931   | 2627  | 7855 | 7567  | 264.54| 6.17 |
|     | Average        | 1000| 951   | 2636  | 7869 | 7660  | 5.22  |
| 11  | S.10.20.150a   | 1594| 1408  | 3959  | 11,949| 11,380| 74.94 | 5.23 |
| 12  | S.10.20.150b   | 1583| 1351  | 4164  | 11,583| 11,269| 114.57| 5.43 |
| 13  | S.10.20.150c   | 1616| 1306  | 4500  | 11,663| 11,315| 124.68| 6.06 |
| 14  | S.10.20.150d   | 1390| 1229  | 3691  | 11,623| 11,330| 173.80| 5.55 |
| 15  | S.10.20.150e   | 1561| 1284  | 4099  | 11,984| 11,596| 253.63| 5.72 |
|     | Average        | 1549| 1316  | 4083  | 11,760| 11,378| 5.60  |
| 16  | S.10.20.200a   | 2176| 1815  | 5177  | 15,790| 15,306| 73.71 | 5.49 |
| 17  | S.10.20.200b   | 2110| 2011  | 4105  | 15,866| 15,642| 117.37| 5.59 |
| 18  | S.10.20.200c   | 2176| 2008  | 3687  | 15,589| 15,257| 124.87| 5.66 |
| 19  | S.10.20.200d   | 1998| 1856  | 4124  | 15,361| 14,886| 174.47| 5.26 |
| 20  | S.10.20.200e   | 2209| 1966  | 3883  | 16,133| 15,879| 259.83| 6.46 |
|     | Average        | 2134| 1931  | 4195  | 15,748| 15,394| 5.69  |
Table 3  Computational results for Set2 instances (Run time: 7,200s)

| No. | Instance   | SFD-R |         |         |         | SFD  |         |         |         |
|-----|------------|-------|---------|---------|---------|------|---------|---------|---------|
|     |            | LB    | UB      | MIPs    | FCuts   | %Gap | MIPs    | FCuts   | %Gap    |
| 1   | L.10.20.50a| 23.38 | 24.46   | 44,629  | 436     | 4.42 | 39,092  | 378     | 10.37   |
| 2   | L.10.20.50b| 42.78 | 44.52   | 44,524  | 418     | 3.92 | 38,201  | 358     | 9.29    |
| 3   | L.10.20.50c| 108.68| 113.33  | 49,221  | 465     | 4.10 | 41,928  | 399     | 10.17   |
| 4   | L.10.20.50d| 170.66| 178.73  | 46,326  | 436     | 4.52 | 39,246  | 375     | 9.62    |
| 5   | L.10.20.50e| 281.98| 293.83  | 45,208  | 432     | 4.03 | 42,330  | 399     | 8.48    |
|     | Average    | 45,982| 437     | 521     | 536     | 5.51 | 42,970  | 443     | 10.25   |
| 6   | L.10.20.100a| 77.13 | 81.63   | 53,231  | 536     | 5.11 | 42,293  | 426     | 9.26    |
| 7   | L.10.20.100b| 107.74| 114.80  | 49,838  | 518     | 6.16 | 43,508  | 451     | 10.16   |
| 8   | L.10.20.100c| 124.70| 132.85  | 47,921  | 492     | 6.14 | 42,553  | 438     | 10.56   |
| 9   | L.10.20.100d| 159.84| 171.03  | 49,894  | 520     | 6.55 | 43,069  | 450     | 10.57   |
| 10  | L.10.20.100e| 264.54| 282.55  | 52,420  | 541     | 6.37 | 43,428  | 452     | 10.68   |
|     | Average    | 50,661| 521     | 50,661  | 536     | 6.42 | 42,970  | 443     | 10.25   |
| 11  | L.10.20.150a| 74.94 | 80.78   | 50,805  | 540     | 7.23 | 43,254  | 454     | 10.11   |
| 12  | L.10.20.150b| 114.57| 123.53  | 48,282  | 512     | 7.25 | 46,069  | 477     | 9.96    |
| 13  | L.10.20.150c| 124.68| 134.85  | 48,507  | 512     | 7.54 | 46,333  | 480     | 10.83   |
| 14  | L.10.20.150d| 173.80| 186.19  | 55,405  | 588     | 6.65 | 44,526  | 475     | 10.60   |
| 15  | L.10.20.150e| 253.63| 274.31  | 50,060  | 534     | 7.54 | 43,369  | 458     | 10.84   |
|     | Average    | 50,612| 537     | 50,612  | 534     | 7.24 | 44,710  | 469     | 10.47   |
| 16  | L.10.20.200a| 73.71 | 79.54   | 51,622  | 562     | 7.33 | 47,067  | 505     | 10.56   |
| 17  | L.10.20.200b| 117.37| 126.17  | 53,803  | 583     | 6.97 | 45,553  | 488     | 9.72    |
| 18  | L.10.20.200c| 124.87| 136.35  | 49,600  | 535     | 8.42 | 45,593  | 483     | 10.51   |
| 19  | L.10.20.200d| 174.47| 188.15  | 50,598  | 555     | 7.27 | 44,756  | 483     | 10.18   |
| 20  | L.10.20.200e| 259.83| 280.19  | 55,014  | 591     | 7.27 | 45,250  | 482     | 10.64   |
|     | Average    | 52,127| 565     | 52,127  | 565     | 7.45 | 45,644  | 488     | 10.32   |
Table 4  Computational results for Set3 instances (Run time: 7,200s)

| No. | Instance     | SFD-R         | SFD | CPLEX         |
|-----|--------------|---------------|-----|---------------|
|     |              | LB            | UB  | MIPs | FCuts | %Gap | MIPs | FCuts | %Gap |
| 1   | L.20.30.500a | 8,693.52      | 8,728.01 | 30,609 | 488  | 0.40 | 25,273 | 369 | –    |
| 2   | L.20.30.500b | 9,804.70      | 9,844.82 | 33,052 | 804  | 0.41 | 29,764 | 644 | 1.80 |
| 3   | L.20.30.500c | 12,862.22     | 12,879.43 | 39,575 | 675  | 0.13 | 37,310 | 627 | 3.27 |
| 4   | L.20.30.500d | 11,380.80     | 11,381.46 | 49,286 | 1,213 | 0.01 | 41,375 | 1022 | 1.39 |
| 5   | L.20.30.500e | 9,443.89      | 9,469.53 | 38,198 | 1,486 | 0.27 | 37,047 | 1296 | –    |
|     | Average      | 38,144        | 933   | 0.24 | 34,154 | 792 | –    |
| 6   | L.20.30.1000a| 9,081.08      | 9,147.39 | 32,568 | 1008 | 0.73 | 31,435 | 979 | 5.42 |
| 7   | L.20.30.1000b| 9,686.60      | 9,718.12 | 41,915 | 768  | 0.33 | 35,272 | 607 | 11.16 |
| 8   | L.20.30.1000c| 10,876.92     | 10,913.56 | 36,185 | 823  | 0.34 | 33,139 | 704 | 12.18 |
| 9   | L.20.30.1000d| 11,706.07     | 11,750.82 | 35,521 | 557  | 0.38 | 33,884 | 458 | –    |
| 10  | L.20.30.1000e| 7,986.09      | 8,001.29 | 35,574 | 597  | 0.19 | 34,962 | 579 | –    |
|     | Average      | 36,353        | 751   | 0.39 | 33,738 | 665 | –    |
| 11  | L.20.30.1500a| 9,295.19      | 9,330.13 | 29,370 | 383  | 0.38 | 27,801 | 350 | –    |
| 12  | L.20.30.1500b| 9,374.68      | 9,390.72 | 31,618 | 510  | 0.17 | 28,794 | 441 | –    |
| 13  | L.20.30.1500c| 9,294.93      | 9,302.69 | 45,199 | 695  | 0.08 | 39,789 | 553 | 32.68 |
| 14  | L.20.30.1500d| 11,618.10     | 11,621.59 | 54,604 | 1351 | 0.03 | 50,537 | 1,127 | 8.01 |
| 15  | L.20.30.1500e| 11,747.13     | 11,767.37 | 38,950 | 658  | 0.17 | 38,072 | 619 | 60.32 |
|     | Average      | 39,948        | 719   | 0.17 | 36,999 | 618 | –    |
| 16  | L.20.30.2000a| 11,477.27     | 11,501.34 | 32,862 | 545  | 0.21 | 31,351 | 467 | –    |
| 17  | L.20.30.2000b| 11,018.40     | 11,047.94 | 28,706 | 483  | 0.27 | 21,206 | 351 | 25.64 |
| 18  | L.20.30.2000c| 9,730.49      | 9,740.92 | 27,570 | 461  | 0.11 | 25,052 | 400 | –    |
| 19  | L.20.30.2000d| 10,664.70     | 10,714.83 | 31,326 | 602  | 0.47 | 29,531 | 549 | –    |
| 20  | L.20.30.2000e| 10,338.18     | 10,396.10 | 29,822 | 639  | 0.56 | 26,951 | 551 | –    |
|     | Average      | 30,057        | 546   | 0.32 | 26,818 | 464 | –    |

Supplementary Information  The online version contains supplementary material available at https://doi.org/10.1007/s10589-023-00457-4.

Data Availability  The data and code that support the findings of this study are available as supplement materials.

Declarations

Conflict of interest  The authors declare no competing interests.

References

1. Ahmed, S., Tawarmalani, M., Sahinidis, N.: A finite branch-and-bound algorithm for two-stage stochastic integer programs. Math. Program. 100(2), 355–377 (2004)
2. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math. Program. 58(1–3), 295–324 (1993)
3. Beier, E., Venkatachalam, S., Corollis, L., Ntaimo, L.: Stage-and scenario-wise fenchel decomposition for stochastic mixed 0–1 programs with special structure. Comput. Oper. Res. 59, 94–103 (2015)
4. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4(1), 238–252 (1962)
5. Birge, John R., Louveaux, Francois: Introduction to stochastic programming. Springer Science & Business Media, New York (2011)
6. Blair, C., Jeroslow, R.: The value function of an integer program. Math. Program. 23(1), 237–273 (1982)
7. Boccia, M., Sforza, A., Sterle, C., Vasilyev, I.: A cut and branch approach for the capacitated p-median problem based on fenchel cutting planes. J Math Model Algorithm 7(1), 43–58 (2008)
8. Boyd, A.E.: Generating fenchel cutting planes for knapsack polyhedra. SIAM J. Optim. 3(4), 734–750 (1993)
9. Boyd, A.E.: Solving integer programs with fenchel cutting planes and preprocessing. In IPCO, pages 209–220, (1993)
10. Boyd, A.E.: Fenchel cutting planes for integer programs. Oper. Res. 42(1), 53–64 (1994)
11. Boyd, A.E.: Solving 0/1 integer programs with enumeration cutting planes. Ann. Oper. Res. 50(1), 61–72 (1994)
12. Boyd, A.E.: On the convergence of fenchel cutting planes in mixed-integer programming. SIAM J. Optim. 5(2), 421–435 (1995)
13. Caroe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Oper. Res. Lett. 24(1–2), 37–45 (1999)
14. Caroe, C.C., Tind, J.: A cutting-plane approach to mixed 0–1 stochastic integer programs. Eur. J. Oper. Res. 101(2), 306–316 (1997)
15. CPLEX. IBM ILOG CPLEX Optimizer. https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/, (2016)
16. Dantzig, G.B., Eaves, C.B.: Fourier-motzkin elimination and its dual. J Combinatorial Theory, Series A 14(3), 288–297 (1973)
17. Gade, Dinakar, Küçükyavuz, Simge, Sen, Suvaject: Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. Math. Program. 144(1–2), 39–64 (2014)
18. Haneveld, W.K.K., Stougie, L., van der Vlerk, M.H.: On the convex hull of the simple integer recourse objective function. Ann. Oper. Res. 56(1), 209–224 (1995)
19. Haneveld, W.K.K., Stougie, L., van der Vlerk, M.H.: An algorithm for the construction of convex hulls in simple integer recourse programming. Ann. Oper. Res. 64(1), 67–81 (1996)
20. Haneveld, W.K.K., van der Vlerk, M.H.: Stochastic integer programming: general models and algorithms. Ann. Oper. Res. 85, 39–57 (1999)
21. Hemmecke, R., Schultz, R.: Decomposition of test sets in stochastic integer programming. Math. Program. 94(2), 323–341 (2003)
22. Kiwiel, K.: Proximity control in bundle methods for convex nondifferentiable minimization. Math. Program. 46, 105–122 (1990)
23. Kong, N., Schaefer, A.J., Hunsaker, B.: Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach. Math. Program. 108(2), 275–296 (2006)
24. Küçükyavuz, Simge, Sen, Suvaject.: An introduction to two-stage stochastic mixed-integer programming. In Leading Developments from INFORMS Communities, pages 1–27. INFORMS, (2017)
25. Laporte, G., Louveaux, F.: The integer L-shaped method for stochastic integer programs with complete recourse. Oper. Res. Lett. 13(3), 133–142 (1993)
26. Louveaux, F.V., Schultz, R.: Stochastic integer programming. In A. Ruszczyński A. Shapiro, editors, Stochastic Programming, volume 10 of Handbooks in Operations Research and Management Science, pages 213 – 266. Elsevier, (2003)
27. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. Wiley-Interscience (1999)
28. Ntaimo, L.: Fenchel decomposition for stochastic mixed-integer programming. J. Global Optim. 55(1), 141–163 (2013)
29. Ntaimo, Lewis: Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. Oper. Res. 58(1), 229–243 (2010)
30. Gurobi Optimization. "gurobi optimizer reference manual," 2014. http://www.gurobi.com, (2014)
31. Ramos, M.T., Sáez, J.: Solving capacitated facility location problems by fenchel cutting planes. J Oper Res Soc 56(3), 297–306 (2005)
32. Römisch, Werner, Vigerske, Stefan: Recent progress in two-stage mixed-integer stochastic programming with applications to power production planning. In Handbook of power systems I, pages 177–208. Springer, (2010)
33. Sáez, J.: Solving linear programming relaxations associated with lagrangean relaxations by fenchel cutting planes. Eur. J. Oper. Res. 121(3), 609–626 (2000)
34. Schultz, R.: Continuity properties of expectation functions in stochastic integer programming. Math. Oper. Res. 18(3), 578–589 (1993)
35. Schultz, R., Stougie, L., van der Vlerk, M.H.: Two-stage stochastic integer programming: a survey. Stat. Neerl. 50(3), 404–416 (1996)
36. Schultz, R., Stougie, L., van der Vlerk, M.H.: Solving stochastic programs with integer recourse by enumeration: a framework using grüber basis. Math. Program. 83(1–3), 229–252 (1998)
37. Sen, S., Higle, J.L.: The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: set convexification. Math. Program. 104(1), 1–20 (2005)
38. Sen, S., Sherali, H.D.: Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. Math. Program. 106(2), 203–223 (2006)
39. Sen, Suveejaeet: Algorithms for stochastic mixed-integer programming models. Handbooks Oper. Res. Management Sci. 12, 515–558 (2005)
40. Sherali, Hanif D., Zhu, Xiaomei: On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. Math. Program. 108(2–3), 597–616 (2006)
41. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, vol. 31. Springer, New York (1998)
42. Sherali, H.D., Fraticelli, B.M.: A modification of benders’ decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. J. Global Optim. 22(1–4), 319–342 (2002)
43. van der Vlerk, M.H.: Convex approximations for complete integer recourse models. Math. Program. 99(2), 297–310 (2004)
44. Van Slyke, R.M., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J. Appl. Math. 17(4), 638–663 (1969)
45. Venkatachalam, S.: algorithms for stochastic integer programs using fenchel cutting planes. PhD thesis, Department of Systems and Industrial Engineering, Texas A &M University, USA, 2014. Available electronically from http://oaktutr.library.tamu.edu/handle/1969.1/153602
46. Venkatachalam, Saravanan, Narayan, Arunachalam: Two-stage absolute semi-deviation mean-risk stochastic programming: an application to the supply chain replenishment problem. Comput Oper Res 106, 62–75 (2019)
47. Vollmer, R.D.: Two stage linear programming under uncertainty with 0–1 integer first stage variables. Math. Program. 19(1), 279–288 (1980)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.