

Integer L-Shaped Method with Non-Supporting No-Good Optimality Cuts

Benjamin P. Riley¹, Prodromos Daoutidis^{*1}, and Qi Zhang^{†1}

¹*Department of Chemical Engineering and Materials Science,
University of Minnesota, Minneapolis, MN 55455, USA*

Abstract

Two-stage stochastic mixed-integer linear programs with mixed-integer recourse arise in many practical applications but are computationally challenging due to their large size and the presence of integer decisions in both stages. The integer L-shaped method with alternating cuts is a widely used decomposition algorithm for these problems, relying on optimality cuts generated using subproblems to iteratively refine the master problem. A key computational bottleneck in this approach is solving the mixed-integer subproblems to optimality in order to generate separating cuts. This work proposes a modification to the integer L-shaped method with alternating cuts to allow for efficient generation of no-good optimality cuts that are separating for the current master problem solutions without being supporting hyperplanes of the feasible region. These separating cuts are derived from subproblems that are terminated before the optimal solution is found or proven to be optimal, reducing the computational effort required for cut generation. Additionally, an updated optimality cut generation function is proposed to account for the various complexities introduced by this early termination strategy. The effectiveness of the proposed method is demonstrated through two case studies on industrially relevant problems from the literature, which illustrate its advantages in handling large-scale instances with complex mixed-integer subproblems. In these cases, the method achieves substantial reductions in solution time or optimality gap compared to the standard integer L-shaped method with alternating cuts, with performance improvements that increase with mixed-integer subproblem size and complexity.

Keywords: two-stage stochastic programming, integer L-shaped method, mixed-integer recourse, stochastic integer programming, decomposition, alternating cuts.

^{*}Corresponding author (daout001@umn.edu)

[†]Primary corresponding author (qizh@umn.edu)

1 Introduction

In this work, we consider two-stage stochastic mixed-integer linear programs (MILPs) of the following form:

$$\begin{aligned}
& \text{minimize} && c^\top x + \hat{c}^\top z + Q(x) \\
& \text{subject to} && Ax + \hat{A}z \geq b \\
& && x \in \{0, 1\}^n \\
& && z \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2},
\end{aligned} \tag{OP}$$

where the expected value of the optimal second-stage cost, $Q(x) := \sum_{s \in \mathcal{S}} p_s Q_s(x)$, is defined via the following optimization problems:

$$\begin{aligned}
Q_s(x) := \min && h_s^\top y_s \\
& \text{s.t.} && F_s x + G_s y_s \geq d_s \\
& && y_s \in \mathbb{R}^{p_1} \times \mathbb{Z}^{p_2}.
\end{aligned} \tag{1}$$

Here, the first-stage decision variables consist of binary state variables (i.e., first-stage variables that affect the second-stage cost), x , and mixed-integer variables that do not directly affect the second stage, z . Additionally, the second-stage decision variables for scenario s , y_s , are mixed-integer.

For ease of exposition, we assume that (OP) is feasible with relatively complete recourse (i.e., problem (1) is feasible for all feasible x in (OP)). When these models are combined, the following deterministic equivalent of the two-stage stochastic program is obtained:

$$\begin{aligned}
& \text{minimize} && c^\top x + \hat{c}^\top z + \sum_{s \in \mathcal{S}} p_s h_s^\top y_s \\
& \text{subject to} && Ax + \hat{A}z \geq b \\
& && F_s x + G_s y_s \geq d_s \quad \forall s \in \mathcal{S} \\
& && x \in \{0, 1\}^n \\
& && z \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2} \\
& && y_s \in \mathbb{R}^{p_1} \times \mathbb{Z}^{p_2} \quad \forall s \in \mathcal{S}.
\end{aligned} \tag{P}$$

One of the most popular decomposition methods for two-stage stochastic programs is Benders decomposition (also known as the L-shaped method). However, it cannot be directly applied to problems of the form (P) due to the integrality constraints on some second-stage decisions. As such, various extensions of Benders decomposition have been developed to solve such problems. For instance, alternative branch-and-cut schemes that branch on second-stage variables, y_s , have been devised to allow Benders decomposition to be applied to the solution of (P) [1, 2]. In [3], the concept of local cuts and global cuts are introduced and used in a novel branch-and-cut scheme to apply Benders cuts to the solution of (P). In [4], concepts from the reformulation-linearization technique and lift-and-project cuts are adapted to sequentially form convex hull representations of the subproblems, making the generated Benders cuts tight despite the presence of second-stage

integrality constraints. In [5], the L-shaped method for continuous recourse is adapted to problems with integer recourse using nonlinear cuts derived from general duality theory.

Several alternative algorithms for the solution of (P) that do not extend the ideas of Benders decomposition have also been proposed. For example, [6] applies a disjunctive decomposition scheme to generate valid inequalities from the solution of multiple relaxations of the second-stage problem. These ideas are extended in [7], where branch-and-cut schemes are incorporated to solve second-stage problems. In [8], an efficient enumeration scheme is proposed for a subset of problems in (P) to generate a set of solutions containing the optimal solution.

Another set of applicable algorithms are decomposition schemes that can be applied to (P) despite not being specifically designed for two-stage stochastic programs. For instance, column generation schemes for mixed-integer optimization problems with complicating constraints have been applied to two-stage stochastic programs by treating non-anticipativity constraints as complicating constraints [9]. This concept has been further developed for the solution of multi-stage stochastic mixed-integer nonlinear programs with discrete state variables [10]. Another algorithm for the solution of multi-stage stochastic mixed-integer programs with discrete state variables is the stochastic dual dynamic integer programming (SDDiP) algorithm, which applies Lagrangian cuts and strengthened Benders cuts to approximate nonconvex cost-to-go functions [11].

The integer L-shaped method of Laporte and Louveaux [12] is a widely used and extensively discussed algorithm that is specifically tailored to exploit the structure of (P). This algorithm has been further developed in numerous works. For instance, [13] extends the method to consider convex nonlinear constraints and incorporate Lagrangian cuts. In several other works, problem-specific features are applied to the algorithm to modify branching schemes [14] or efficiently solve subproblems to generate cuts [15]. In [16], a problem-specific dynamic programming algorithm for vehicle routing is applied to the integer L-shaped method to solve subproblems efficiently. Similarly, [17] applies the same problem-specific algorithm for solving vehicle routing subproblems to generate additional problem-specific cuts.

Despite these successful applications of the integer L-shaped method where problem-specific features are used, there is a need for further development of the algorithm and related algorithms, as many industrially-relevant stochastic programs are large and lack convenient problem-specific features. This work aims to further develop the integer L-shaped method to improve algorithm scalability and increase the number of problems the algorithm is well-suited for. Here, we specifically focus on an extension of the *integer L-shaped method with alternating cuts* of Angulo, Ahmed, and Dey [18]. In this algorithm, Benders cuts that are violated by the current master problem solution are sought before computationally costly mixed-integer subproblems are solved.

Following this conceptually simple yet effective strategy, we propose a further modification to the algorithm to allow for efficient generation of no-good optimality cuts that are separating for current master problem solutions without being supporting hyperplanes of the feasible region (i.e., tight cuts). This is achieved by terminating mixed-integer subproblems before the optimal solution is found or proven to be optimal. Additionally, an updated optimality cut generation function is proposed to account for the various complexities introduced by terminating subproblems early. Two computational case studies are presented to demonstrate the potential advantages provided by the proposed modification.

The remainder of this paper is organized as follows. In Section 2, we describe the integer

L-shaped method with alternating cuts from the literature. Section 3 describes the proposed modification to the algorithm and the corresponding updated optimality cut generation function. Sections 4.1 and 4.2 focus on the case studies on modular, relocatable manufacturing units in a supply chain and the design of a renewables-based fuel and power production network, respectively. Finally, we provide concluding remarks in Section 5.

2 Integer L-Shaped Method with Alternating Cuts

The following is a multicut implementation of the *integer L-shaped method with alternating cuts* [18] for two-stage stochastic MILPs with binary state variables and mixed-integer recourse variables. The algorithm generates so-called optimality cuts in parallel to form convex underapproximations of the optimal value functions of the second-stage scenarios, sometimes referred to as recourse functions, $Q_s(x)$.

The algorithm works by solving a reformulation of (P) referred to as the master problem (MP):

$$\text{minimize} \quad c^\top x + \hat{c}^\top z + \sum_{s \in \mathcal{S}} p_s \eta_s \quad (2)$$

$$\text{subject to} \quad Ax + \hat{A}z \geq b \quad (3)$$

$$e_{sk}^\top x + \eta_s \geq f_{sk} \quad \forall s \in \mathcal{S}, k \in \mathcal{K}_s \quad (4)$$

$$x \in \{0, 1\}^n \quad (5)$$

$$z \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2} \quad (6)$$

$$\eta \in \mathbb{R}_+^S, \quad (7)$$

where constraints (4) are optimality cuts that restrict η_s to have a value greater than or equal to $Q_s(x)$ for any feasible x .

For the reformulation to be exact, each cut must satisfy $Q_s(x) \geq f_{sk} - e_{sk}^\top x$ for all feasible x (i.e., the cuts underestimate $Q_s(x)$). Additionally, for each s and feasible x^* , there must exist a constraint in (4) such that $Q_s(x^*) = f_{sk} - e_{sk}^\top x^*$ (i.e., at least one underestimator must be tight at x^*). As seen commonly in the literature [2, 18], the generation of these optimality cuts is incorporated in a branch-and-cut framework for the solution of (MP). During the solution of (MP), the formulation is relaxed, and (4) starts as an empty set of constraints. Optimality cuts are then added iteratively via an optimality cut generation function. The cut generation function involves solving $S = |\mathcal{S}|$ subproblems derived from the reformulation of (P). If x were to take a fixed value, (P) would be separable into S separate MILPs. These MILPs define the optimal value functions of the second-stage scenarios, $Q_s(x)$, presented above. They will be referred to as *mixed-integer subproblems* in the remainder of this work:

$$\begin{aligned} Q_s(x) &:= \min \quad h_s^\top y_s \\ \text{s.t.} \quad &F_s x + G_s y_s \geq d_s \\ &y_s \in \mathbb{R}^{p_1} \times \mathbb{Z}^{p_2}. \end{aligned} \quad (\text{MISP}_s(x))$$

The dual programs of the LP relaxations of the mixed-integer subproblems are referred to as

the dual separation problems:

$$\begin{aligned} Q_s^{\text{LP}}(x) := \max \quad & (d_s - F_s x)^\top u \\ \text{s.t.} \quad & G_s^\top u = h_s \\ & u \in \mathbb{R}_+^m, \end{aligned} \tag{DSP}_s(x)$$

where $Q_s^{\text{LP}}(x) \leq Q_s(x)$. The first of the two types of optimality cuts that are generated by the algorithm and included in (4) are *Benders optimality cuts*,

$$\eta_s \geq (d_s - F_s x)^\top u_t \quad \forall t \in \mathcal{T}_s, \tag{8}$$

where u_t are the extreme points of the feasible region for (DSP_s). Benders optimality cuts are generated by solving (DSP_s(x)) and using the optimal extreme point u^* to generate a constraint of the form (8). These constraints generally do not provide a tight underestimate of $Q_s(x)$ at any point and therefore are not supporting hyperplanes of the feasible region. Regardless, the constraints can be used to enforce valid lower bounds on η_s and improve the formulation.

The second type of optimality cuts that are generated by the algorithm and included in (4) are referred to as *no-good optimality cuts* in this work:

$$\eta_s \geq Q_s(x^*) + (L_s - Q_s(x^*)) \left(\sum_{j:x_j^*=0} x_j + \sum_{j:x_j^*=1} (1 - x_j) \right). \tag{9}$$

Here, L_s is a lower bound on $Q_s(x)$ for all x values that are feasible in (P). One valid value for L_s is the optimal objective value of

$$\begin{aligned} \underset{x, y_s}{\text{minimize}} \quad & h_s^\top y_s \\ \text{subject to} \quad & F_s x + G_s y_s \geq d_s \\ & x \in \{0, 1\}^n \\ & y_s \in \mathbb{R}^{p_1} \times \mathbb{Z}^{p_2}. \end{aligned} \tag{LBP}_s$$

No-good optimality cuts are generated by solving (MISP_s(x)) with x as the current solution to (MP), x^* , and using the optimal objective value to generate a separating constraint of the form (9). A constraint is said to be *separating* if the constraint is satisfied by all feasible solutions to (P) but is violated by the current solution to the relaxation of (MP). Unlike Benders optimality cuts, constraints (9) provide tight underestimates of $Q_s(x)$ at x^* and are therefore supporting hyperplanes of the feasible region of (MP).

A general branch-and-cut implementation of the integer L-shaped method is presented in Algorithm 1. When used with the standard optimality cut generation function, described in Algorithm 2, the method is referred to as the integer L-shaped method with alternating cuts [18]. The algorithm proceeds like a standard branch-and-cut procedure by solving multiple relaxations of the master problem with branching constraints while establishing lower and upper bounds to prune or fathom nodes from the tree. The algorithm does this via the following changes to the typical branch-and-cut algorithm. First, integer-feasible nodes that are not fathomed by infeasibility

or bound are not immediately accepted as a new incumbent solution and fathomed. Instead, separating Benders and no-good optimality cuts are generated by solving $(\text{DSP}_s(x))$ and $(\text{MISP}_s(x))$ for all scenarios s and then added to (MP). A certificate of feasibility for the current solution is obtained only if no separating cuts are generated, at which point the upper bound and incumbent solution are updated, and the node is fathomed. Second, before branching is done at the root node, separating Benders cuts are sought at fractional solutions to the root node. One valid way of interpreting this step is that the LP relaxation of (P) is first solved via Benders decomposition.

Algorithm 1 General Integer L-shaped method

Input: $c, \hat{c}, p, A, \hat{A}, b, \mathcal{S}, \epsilon$

Output: Optimal solution $\tilde{x}$ to (P) and optimal value UB

```

1: Solve  $(\text{LBP}_s)$  to compute  $L_s$  for all  $s \in \mathcal{S}$ 
2: Initialize the branch-and-bound tree for solving (MP) by adding the root relaxation to the list
   of leaf nodes. Set  $UB$  and  $LB$  to appropriate initial bounds
3: while  $\frac{|UB-LB|}{|UB|} > \epsilon$  do
4:   Select a node from the list of leaf nodes
5:   Solve the relaxation of (MP) at the current node
6:   if the current node is infeasible then
7:     Fathom the node and return to line 4
8:   Record the optimal solution  $x^*, z^*$ , and  $\eta^*$ . Following the usual branch and cut procedure,
     update the lower bound of the tree,  $LB$ , if necessary
9:   if the current node is the root relaxation then
10:    Call the cut generation function (Algorithm 2) with  $x^*$ 
11:    if a cut was generated then
12:      Add generated cuts to (MP) and return to line 4
13:  if  $c^\top x^* + \hat{c}^\top z^* + \sum_{s \in \mathcal{S}} p_s \eta_s^* > UB$  then
14:    Fathom the node and return to line 4
15:  if  $(x^*, z^*)$  contains fractional components that violate integrality constraints then
16:    Select one to branch on following the usual branch-and-cut procedure
17:    Remove the current node and append the new nodes to the list of leaf nodes
18:  else
19:    Call the cut generation function (Algorithm 2) with  $x^*$   $\triangleright UB, \tilde{x}$  updated
20:    if a cut was generated then
21:      Add generated cuts to (MP)
22:    else
23:      Fathom the node

```

While the typical version of the algorithm described in this section generates Benders cuts at fractional solutions to the root relaxation, the algorithm could instead be configured differently. For instance, branching could occur immediately at the root relaxation and all cut generation could be performed at integer-feasible node solutions. Alternatively, separating Benders cuts could be sought at fractional solutions before branching at all nodes in the tree, rather than just the root node, as described in [2]. This work includes the implementation that appears most commonly in the literature.

The standard optimality cut generation function, described in Algorithm 2, uses an *alternating*

cut strategy. In this strategy, dual separation problems are solved first to generate Benders cuts. If any separating Benders cuts are found, the cut generation function returns those cuts and terminates; otherwise, the algorithm proceeds to solve mixed-integer subproblems to generate no-good optimality cuts. This is done to reduce computational effort, as the separating Benders cuts may be sufficient to advance the solution of the master problem—either by fathoming the node by bound or by raising the lower bound of the node sufficiently so that another leaf node is selected next (in this instance, the original node may be fathomed by bound when the upper bound is subsequently lowered). The performance improvement from this strategy may be large, as each $(\text{DSP}_s(x))$ is an LP, whereas each $(\text{MISP}_s(x))$ is a potentially NP-hard MILP. Additionally, the optimality cut generation function keeps lists of x values, V and V^{LP} , for which the subproblems have already been solved to avoid duplicate evaluations.

Algorithm 2 Standard multicut optimality cut generation function

Input: Current (MP) solution (x^*, z^*, η^*) , V , V^{LP}

Output: A (potentially empty) set of separating cuts for the current (MP) solution: $\mathcal{C}$

```

1: Initialize  $\mathcal{C} = \emptyset$ 
2: if  $x^* \in V$  then
3:   return  $\mathcal{C}$   $\triangleright$  We know  $\eta_s^* \geq Q_s(x^*) \quad \forall s \in \mathcal{S}$ 
4: if  $x^* \notin V^{\text{LP}}$  then
5:   for  $s \in \mathcal{S}$  do  $\triangleright$  Can be parallelized
6:     Solve  $(\text{DSP}_s(x^*))$  and record optimal dual solution vector  $u^*$ 
7:     if  $(d_s - F_s x^*)^\top u^* > \eta_s^*$  then
8:        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\eta_s \geq (d_s - F_s x)^\top u^*\}$ 
9:    $V^{\text{LP}} \leftarrow V^{\text{LP}} \cup \{x^*\}$ 
10:  if  $\mathcal{C} \neq \emptyset$  then
11:    return  $\mathcal{C}$ 
12: if  $x^* \in \{0, 1\}^n$  then
13:   for  $s \in \mathcal{S}$  do  $\triangleright$  Can be parallelized
14:     Solve  $(\text{MISP}_s(x^*))$  and record optimal value  $Q_s(x^*)$ 
15:     if  $Q_s(x^*) > \eta_s^*$  then
16:        $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ \eta_s \geq Q_s(x^*) + (L_s - Q_s(x^*)) \left( \sum_{j: x_j^* = 0} x_j + \sum_{j: x_j^* = 1} (1 - x_j) \right) \right\}$ 
17:   if  $c^\top x^* + \hat{c}^\top z^* + \sum_{s \in \mathcal{S}} p_s Q_s(x^*) < UB$  then
18:      $UB \leftarrow c^\top x^* + \hat{c}^\top z^* + \sum_{s \in \mathcal{S}} p_s Q_s(x^*)$ 
19:      $\tilde{x} \leftarrow x^*$ 
20:    $V \leftarrow V \cup \{x^*\}$ 
21: return  $\mathcal{C}$ 

```

3 Integer L-Shaped Method with Non-Supporting No-Good Optimality Cuts

The Benders cuts in the integer L-shaped method with alternating cuts provide a fast avenue for generating separating cuts without solving an MILP. As discussed above, these cuts may be sufficient for the solution of (MP), and the algorithm may never generate the corresponding no-good optimality cut at the same x^* . As detailed in [18], this modification yields speedups of one order of magnitude for some problem instances from the literature.

Consider instances where the lower bound on η_s at x^* needed for the advancement of the algorithm is greater than $Q_s^{\text{LP}}(x^*)$ (and therefore cannot be enforced by a Benders cut) but significantly less than $Q_s(x^*)$. In these instances, the integer L-shaped method with alternating cuts would not be able to efficiently generate a separating cut and would instead need to generate a cut of form (9). This is particularly costly in the context of two-stage stochastic programming, where subproblems are parallelized on a per-scenario basis. In this context, all parallel workers must wait for the last subproblem to be solved before proceeding to the next step. Thus, a single particularly difficult ($\text{MISP}_s(x)$) can lead to a significantly diminished performance advantage from parallelization. In this section, we propose a conceptually simple modification to the algorithm that allows for the efficient generation of cuts that enforce lower bounds between $Q_s^{\text{LP}}(x^*)$ and $Q_s(x^*)$.

The new cuts used by the modified algorithm are, again, no-good optimality cuts:

$$\eta_s \geq \underline{Q}_s(x^*) + (L_s - \underline{Q}_s(x^*)) \left(\sum_{j:x_j^*=0} x_j + \sum_{j:x_j^*=1} (1 - x_j) \right). \quad (10)$$

Cuts (10) differ from (9) in that they use a general lower bound on $Q_s(x^*)$, $\underline{Q}_s(x^*)$, in place of $Q_s(x^*)$ itself. Unlike (9), cuts (10) generally do not provide a tight underestimate of $Q_s(x)$ at any point and therefore are not supporting hyperplanes of the feasible region. In this sense, they serve the same purpose as Benders cuts in the algorithm: enforce valid bounds on η_s that can be generated efficiently.

In the proposed algorithm, the lower bounds, $\underline{Q}_s(x^*)$, are generated by solving ($\text{MISP}_s(x)$) to a termination criteria other than proof of optimality. In this work, the termination criteria considered are terminal optimality gap and time limit. These termination criteria are gradually loosened (i.e., the time limit is extended or the terminal optimality gap is lessened) to generate sequentially tighter cuts as needed—if necessary, the supporting hyperplane of (9) is eventually obtained. The proposed algorithm uses the same general integer L-shaped method of Algorithm 1 but uses the new optimality cut generation function Algorithm 3 instead of Algorithm 2.

One major change incorporated into the modified optimality cut generation function is the accounting of previous subproblem solutions. In particular, functions $l : V \rightarrow \mathbb{N}$ and $\bar{t} : V \rightarrow \mathbb{R}$ are added to keep a record of the number of times ($\text{MISP}_s(x)$) has been solved and the current time limit of the subproblem at each x , respectively. Further, $l(x)$ is used to determine the terminal optimality gap used when solving ($\text{MISP}_s(x)$). This is done via $(\alpha_k)_{k=1}^K$, a finite, decreasing sequence of terminal optimality gaps with length K that ends in 0. The sequence of optimality gaps is restricted to have these properties so that the algorithm will generate the supporting hyperplanes of (9) (corresponding to an optimality gap of 0) in finite time. As such, the finite convergence of the

Algorithm 3 Optimality cut generation function with non-supporting no-good optimality cuts

Input: Current (MP) solution (x^*, z^*, η^*) , V , V^{LP} , $(\alpha_k)_{k=1}^K$, $\bar{t}^{\text{init}}$.

Output: A (potentially empty) set of separating cuts for the current (MP) solution: $\mathcal{C}$

```

1: Initialize  $\mathcal{C} = \emptyset$ 
2: if  $x^* \in V \wedge l(x^*) = K$  then
3:   return  $\mathcal{C}$   $\triangleright$  We know  $\eta_s^* \geq Q_s(x^*) \quad \forall s \in \mathcal{S}$ 
4: if  $x^* \notin V^{\text{LP}}$  then
5:   for  $s \in \mathcal{S}$  do  $\triangleright$  Can be parallelized
6:     Solve  $(\text{DSP}_s(x^*))$  and record optimal solution vector  $u^*$ 
7:     if  $(d_s - F_s x^*)^\top u^* > \eta_s^*$  then
8:        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\eta_s \geq (d_s - F_s x)^\top u^*\}$ 
9:    $V^{\text{LP}} \leftarrow V^{\text{LP}} \cup \{x^*\}$ 
10:  if  $\mathcal{C} \neq \emptyset$  then
11:    return  $\mathcal{C}$ 
12: if  $x^* \in \{0, 1\}^n$  then
13:   if  $x^* \notin V$  then
14:      $V \leftarrow V \cup \{x^*\}$ 
15:      $l(x^*) \leftarrow 0$ 
16:      $\bar{t}(x^*) \leftarrow \bar{t}^{\text{init}}$ 
17:  while  $l(x^*) < K$  do
18:    for  $s \in \mathcal{S}$  do  $\triangleright$  Can be parallelized
19:      Solve  $(\text{MISP}_s(x^*))$  with terminal optimality gap  $\alpha_{l(x^*)+1}$  and time limit  $\bar{t}(x^*)$ 
20:      Record terminal lower bound  $\underline{Q}_s(x^*)$  and terminal upper bound  $\overline{Q}_s(x^*)$  if one was found
21:      if  $\underline{Q}_s(x) > \eta_s^*$  then
22:         $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ \eta_s \geq \underline{Q}_s(x^*) + (L_s - \underline{Q}_s(x^*)) \left( \sum_{j: x_j^* = 0} x_j + \sum_{j: x_j^* = 1} (1 - x_j) \right) \right\}$ 
23:      if  $\overline{Q}_s(x^*)$  was found for all  $s \in \mathcal{S}$  and  $c^\top x^* + \hat{c}^\top z^* + \sum_{s \in \mathcal{S}} p_s \overline{Q}_s(x^*) < UB$  then
24:         $UB \leftarrow c^\top x^* + \hat{c}^\top z^* + \sum_{s \in \mathcal{S}} p_s \overline{Q}_s(x^*)$ 
25:         $\hat{x} \leftarrow x^*$ 
26:      if Any subproblems terminated due to time limit then
27:         $\bar{t}(x^*) \leftarrow 2\bar{t}(x^*)$ 
28:         $l(x^*) \leftarrow \min\{l(x^*) + 1, K - 1\}$ 
29:      else
30:         $l(x^*) \leftarrow l(x^*) + 1$ 
31:      if  $\mathcal{C} \neq \emptyset$  then
32:        return  $\mathcal{C}$ 
33: return  $\mathcal{C}$ 

```

integer L-shaped method is maintained after the proposed modifications.

Every time a set of subproblems is solved to an optimality gap of $\alpha_{l(x^*)+1}$ for x^* , $l(x^*)$ is increased by 1. This causes subsequent iterations of $(\text{MISP}_s(x^*))$ to be solved to a tighter terminal optimality gap. Additionally, if any subproblems are terminated due to the time limit, $\bar{t}(x^*)$ is doubled. If the time limit terminates a subproblem when the terminal optimality gap is 0, $l(x^*)$ is not incremented so that the subproblem will be executed with a terminal optimality gap of 0 again. While not necessary for the algorithm's function, the incomplete branch-and-cut trees used to solve mixed-integer subproblems to nonzero optimality gaps can be saved and then resumed in subsequent iterations at the same x^* to eliminate duplicated computations. This, however, could be highly memory- or storage-intensive depending on the implementation.

The no-good optimality cuts generated in Algorithm 2 are supporting hyperplanes of the feasible region, meaning that a certificate of feasibility for the current solution is obtained once it is determined that the solution does not violate the cut at x^* . Conversely, the non-supporting optimality cuts generated in Algorithm 3 might not be violated by an infeasible solution to (MP). To address this, a while loop is added to the optimality cut generation function to continue generating tighter cuts when no separating cuts are generated in an iteration. This loop continues while $l(x^*) < K$, indicating that $(\text{MISP}_s(x^*))$ has not been solved to an optimality gap of 0, and terminates when a separating cut is found.

One additional consequence of the changes to the algorithm is that new feasible solutions and potential incumbent solutions are not generated from the lower bounds used in the no-good optimality cuts. Rather, upper bounds on $(\text{MISP}_s(x^*))$, $\bar{Q}_s(x^*)$, are obtained for each $s \in S$ and then combined (in lines 23–25) to construct feasible solutions and potential incumbent solutions to (MP).

4 Case Studies

In this section, the proposed integer L-shaped method with non-supporting no-good optimality cuts is evaluated using two industrially relevant two-stage stochastic programs. In the first case study, the optimal set of investments into modular, relocatable manufacturing units is sought for a supply chain that faces uncertain future demands. In the second case study, an optimal process design is sought for a renewables-based fuel and power production network that faces uncertain power generation profiles.

In both case studies, the problem sizes are modulated and the performance of the proposed algorithm is compared to that of the integer L-shaped method with alternating cuts from the literature and directly solving the full-space problem. Subproblems are solved in parallel in both the integer L-shaped method from the literature and the proposed modified algorithm.

4.1 Case Study 1: Supply Chain with Relocatable Manufacturing Facilities

The problem considered in this case study consists of selecting modular, relocatable manufacturing units for a supply chain facing uncertain future demands. The objective is to invest in a set of modular manufacturing units that yields a minimal combined capital and expected operating cost over the modeled planning horizon. This problem is adapted from [19], in which it is a deter-

ministic problem. The deterministic equivalent of the two-stage stochastic program is presented below:

$$\begin{aligned} \text{minimize} \quad & \sum_{s \in \mathcal{S}} p_s \left(\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{I}} c_{ijt} d_{its} x_{ijts} + \sum_{j' \in \mathcal{J}, m \in \mathcal{M}} h_{jj'mt} w_{jj'mts} \right) \right. \\ & \left. + \sum_{i \in \mathcal{I}, t \in \mathcal{T}} p_{it} q_{its} d_{its} \right) + \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}_1} g_m y_{mk} \end{aligned} \quad (11)$$

$$\text{subject to} \quad y_{mk} \leq y_{m,k-1} \quad \forall m \in \mathcal{M}, k = 2, \dots, |\mathcal{K}_m| \quad (12)$$

$$z_{ms} = \sum_{k \in \mathcal{K}_m} y_{mk} \quad \forall m \in \mathcal{M}, s \in \mathcal{S} \quad (13)$$

$$\sum_{j \in \mathcal{J}} x_{ijts} + q_{its} = 1 \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \quad (14)$$

$$v_{0mts} = z_{ms} + \sum_{t'=1}^t \left(\sum_{j' \in \mathcal{J}} w_{j'jmts} - \sum_{j' \in \mathcal{J}} w_{jj'mts} \right) \quad \forall m \in \mathcal{M}, t \in \mathcal{T}, s \in \mathcal{S} \quad (15)$$

$$v_{jmts} = \sum_{t'=1}^t \left(\sum_{j' \in \mathcal{J}} w_{j'jmts} - \sum_{j' \in \mathcal{J}} w_{jj'mts} \right) \quad \forall j \in \mathcal{J}, m \in \mathcal{M}, t \in \mathcal{T}, s \in \mathcal{S} \quad (16)$$

$$\sum_{i \in \mathcal{I}} d_{its} x_{ijts} \leq \sum_{m \in \mathcal{M}} v_{jmts} u_m \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, s \in \mathcal{S} \quad (17)$$

$$\sum_{i \in \mathcal{I}} d_{its} x_{ijts} \leq u^{\max} \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, s \in \mathcal{S} \quad (18)$$

$$y_{mk} \in \{0, 1\} \quad \forall m \in \mathcal{M}, k \in \mathcal{K}_m \quad (19)$$

$$0 \leq q_{its} \leq 1 \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \quad (20)$$

$$0 \leq x_{ijts} \leq 1 \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, s \in \mathcal{S} \quad (21)$$

$$z_{jms} \in \mathbb{Z}_+ \quad \forall j \in \mathcal{J}, m \in \mathcal{M}, s \in \mathcal{S} \quad (22)$$

$$v_{jmts} \in \mathbb{Z}_+ \quad \forall j \in \mathcal{J}, m \in \mathcal{M}, t \in \mathcal{T}, s \in \mathcal{S} \quad (23)$$

$$w_{jj'mts} \in \mathbb{Z}_+ \quad \forall j \in \mathcal{J}, j' \in \mathcal{J}, m \in \mathcal{M}, t \in \mathcal{T}, s \in \mathcal{S}. \quad (24)$$

In this model, y_{mk} is a binary variable indicating whether or not manufacturing unit k of size m is selected—these are the only first-stage variables in the model. Second-stage variables are as follows: z_{ms} is an auxiliary integer decision that corresponds to the number of units of size m that are selected, x_{ijts} is the fraction of demand at site i that is satisfied by production site j , q_{its} is the fraction of demand at site i that is not delivered, v_{jmts} is the number of units of size m located at site j at time t , and $w_{jj'mts}$ is the number of units of size m that are relocated from production site j to j' at time t . The objective function of the model includes terms for the cost of production, the cost of relocating units, a penalty cost for unmet demand, and the capital cost of the manufacturing units.

Constraints (12) are symmetry-breaking constraints for the binary investment decisions. Constraints (13) relate the binary investment decisions and the auxiliary z_{ms} variables. Constraints

(14) account for the amount of demand at site i that is delivered. Constraints (15) and (16) account for the number of units at each site over time. Finally, constraints (17) and (18) impose bounds on the amount of production that can occur at site i ; the number of manufacturing units present imposes one bound, and the other is a limit on the total production that can occur at any site. Readers are directed to [19] for further discussion of the model.

Several changes were made to the model to make it a two-stage stochastic program that is compatible with the proposed algorithm. First, the demands, d_{its} , were made to be random, scenario-dependent parameters. Second, the integer decisions corresponding to the number of modules that were selected were moved to the second stage as auxiliary variables, and the choice to invest in individual units was instead modeled in the first stage via binary y_{mk} . Additionally, the model was made to have relatively complete recourse by the incorporation of q_{its} and its associated penalty term in the objective function.

The size of the supply chain network and the number of scenarios are varied to create nine different problem sizes. The network size is chosen because it is a key parameter that affects the size and complexity of the subproblems. Then, five problem instances are randomly generated using data from Table 2 in [19] (which includes the distribution for d_{its}) for each of the nine problem sizes. Each instance of each problem size is then solved directly as the given full-space model, using the integer L-shaped method with alternating cuts from the literature, and using the proposed modified integer L-shaped method with non-supporting no-good optimality cuts (shortened to "Mod. Int. L-shaped"). The integer L-shaped methods are implemented with 32 parallel workers for the solution of the subproblems. All optimization instances are solved using Gurobi version 12.0 [20], and all algorithms are implemented in JuMP [21] via the Julia programming language [22]. Additionally, $(\alpha_k) = (0.1, 0.01, 0)$ is selected as the sequence of optimality gaps, and 600 seconds is selected as the initial mixed-integer subproblem time limit. Computational statistics of the investigation are summarized in Table 1.

Table 1: Summary of computational statistics for the case study on the design of supply chains with modular manufacturing units.

$ \mathcal{I} / \mathcal{J} $	$ \mathcal{S} $	Full-space			Int. L-shaped			Mod. int. L-shaped		
		NS	$\overline{\text{gap}}$ (%)	time (s)	NS	$\overline{\text{gap}}$ (%)	time (s)	NS	$\overline{\text{gap}}$ (%)	time (s)
25/5	32	0	-	853	0	-	223	0	-	250
	64	1	0.68	2,089	0	-	409	0	-	468
	96	0	-	2,794	0	-	716	0	-	820
50/10	32	2	0.97	4,492	1	0.57	3,428	0	-	1,900
	64	4	27.05	6,357	2	1.09	4,430	0	-	3,302
	96	5	60.17	-	2	3.20	4,291	0	-	5,101
75/15	32	4	28.95	6,018	5	7.14	-	4	4.43	7,094
	64	5	66.98	-	5	29.59	-	4	4.46	7,561
	96	5	99.96	-	5(1)	9.23 ^a	-	5	3.45	-

^a Numbers in parentheses indicate the number of problem instances where no feasible solution was found. These instances were excluded from the calculation of average optimality gap.

In Table 1, NS stands for "not solved" and indicates the number of problem instances not solved to optimality within the 3-hour time limit. The average optimality gaps of instances that

were not solved to optimality are presented in the $\overline{\text{gap}}$ (%) columns. Further, the average computation times of the instances that were solved to optimality are presented in the **time (s)** column.

As can be seen from the table, all algorithms manage to solve most problem instances with the small, 30-node network to optimality. However, the computation time of the integer L-shaped methods is several times shorter than what is required to directly solve the full-space models for these instances.

For the medium, 60-node network, Gurobi manages to solve 4 of 15 problem instances in full space to optimality while the integer L-shaped method from the literature solves 10 and the proposed method solves all 15. Further, it can be seen that the terminal optimality gap of these instances for the full-space model increases rapidly with increasing scenario count; this is an intuitive result as it does not exploit the decomposable structure of the model.

The advantage of the proposed method over the integer L-shaped method from the literature is most clear from the results for the large, 90-node network. At this network size, the modified integer L-shaped method is able to solve the problem instances to a terminal optimality gap several times less than the standard integer L-shaped method. One potentially misleading statistic is the average gap of 9.23% for the standard algorithm and the largest problem instances, which is the average of the four instances where the algorithm found a feasible solution and upper bound. If this instance were treated as having a terminal gap of 100%, then this statistic would be near 30%, like the instances with 64 scenarios.

To evaluate the difference in the behavior of the proposed and standard integer L-shaped methods, the evolution of the upper and lower bounds of the algorithms while solving a specific problem instance with a 90-node network and 64 scenarios is plotted in Figure 1.

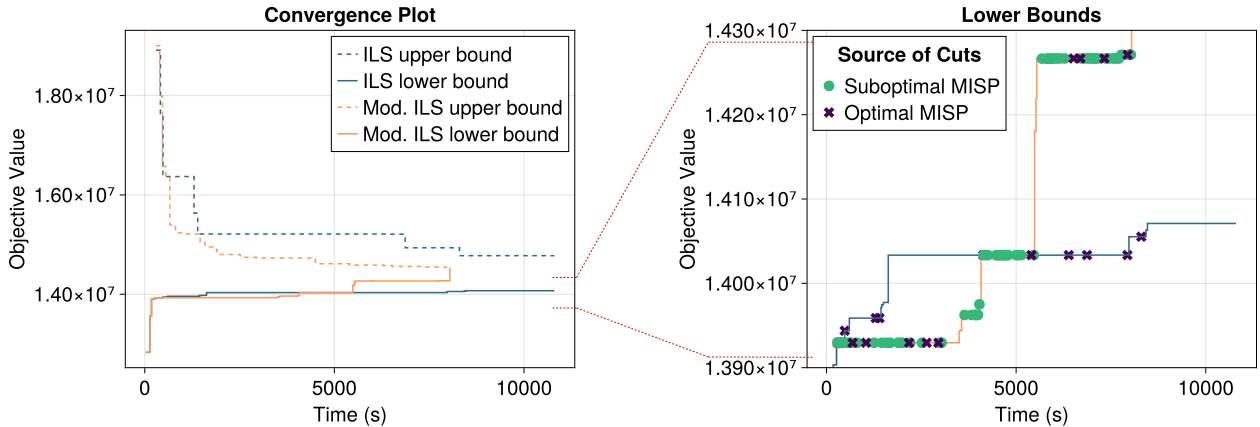


Figure 1: Evolution of upper and lower bounds of the proposed algorithm (Mod. ILS) and standard integer L-shaped method (ILS).

From the figure, it can be observed that the proposed algorithm solves many more mixed-integer subproblems than the standard algorithm. Additionally, it appears that the lower bound of the standard algorithm remains at the same value for a relatively long time between updates. This demonstrates the key difference between the algorithms: the proposed algorithm generates many relatively loose cuts, while the standard algorithm generates fewer, tighter cuts by solving difficult MILP subproblems to optimality. This takes more than an hour in some subproblem instances. The demonstrated trade-off ultimately provides an advantage that scales with the size

and difficulty of the subproblem, as indicated in Table 1.

4.2 Case Study 2: Renewables-Based Fuel and Power Production Network

The problem considered in this case study is a superstructure-based network design optimization problem that seeks to design a minimal-cost fuel and power production network that uses intermittent and uncertain wind and solar power. The problem is adapted from [23], in which it is a deterministic problem. The problem’s superstructure models resources (e.g., hydrogen, syngas, and power) and process units as separate nodes. Process units are modeled using multiple operating modes, where some modes correspond to production and others correspond to startup or shutdown. The model includes constraints on the operating modes of each unit, enforcing restrictions such as minimum stay times in some modes, fixed durations of other modes, and predefined transition sequences between some sets of modes. Additionally, storage is available to store resources, and the accumulation of these resources over the planning horizon is modeled. The planning horizon is separated into multiple “seasons,” and each season is modeled as a scheduling horizon that cycles multiple times. The deterministic equivalent of the two-stage stochastic program used in this case study is presented in the appendix.

Several changes were made to the model to make it a two-stage stochastic program that is compatible with the proposed algorithm. First, the wind and solar power generation profiles were made to be random, scenario-dependent parameters. Second, storage is assumed to be already built and is therefore not a decision variable. Additionally, first-stage binary decisions are introduced that indicate whether or not process units of fixed nameplate capacities are selected. Furthermore, more operating modes are introduced to the process units (for a total of 5 modes), with each mode having an operating range of 20% of the unit capacity. Finally, additional constraints on the set of feasible investment decisions are made to improve the formulation of the master problem. These constraints make it so that a process unit cannot be built unless there is a way to produce all of the intermediate resources used by the unit (i.e., it is not possible to build units that cannot be used).

Similar to the first case study, the number of time periods in the scheduling horizon and the number of scenarios are varied to create nine different problem sizes. The number of time periods is chosen because it is a key parameter that affects the size and complexity of the subproblems. Five problem instances are randomly generated from historical wind and solar irradiation data for each of the nine problem sizes. Then, the same three algorithms are used to solve each problem instance. The same choices of (α_k) and $\bar{t}(x)$ are used as in the first case study. A summary of the computational statistics from this case study is presented in Table 2.

Like in the first case study, all algorithms perform well on the smallest problem instances. For all other problem instances, however, Gurobi fails to solve any full-space models to optimality and terminates with an optimality gap of at least 79%. Clearly, Gurobi does not scale well for the examined problem. The integer L-shaped method from the literature also demonstrates diminished scaling for this problem compared to the first case study, with 7 instances failing to find a feasible solution within the time limit. Additionally, it appears that the performance of the algorithm is highly varied and unpredictable in this case, with some instances being solved to an optimality gap of 10%, and others of the same size that terminate before a feasible solution is found. This further highlights the potential drawback of the standard integer L-shaped method: the algorithm

Table 2: Summary of computational statistics for case study on renewables-based fuel and power production network.

$ \mathcal{T} $	$ \mathcal{S} $	Gurobi			Int. L-shaped			Mod. int. L-shaped		
		NS	gap (%)	time (s)	NS	gap (%)	time (s)	NS	gap (%)	time (s)
24	8	2	0.61	2,963	0	-	3,496	0	-	1,641
	16	5	80.60	-	0	-	9,124	0	-	3,105
	24	5	99.76	-	1	8.81	5,427	0	-	3,871
36	8	5	79.37	-	3	43.04	1,489	0	-	2,094
	16	5	99.99	-	3(2)	10.50 ^a	12,148	0	-	2,418
	24	5	99.87	-	4(2)	97.70 ^a	3,915	1	19.00	5,511
48	8	5	99.99	-	2	65.94	4,807	0	-	5,850
	16	5	99.99	-	3(2)	61.59 ^a	9,535	0	-	8,289
	24	5	99.99	-	2(1)	96.60 ^a	-	0	-	10,449

^a Numbers in parentheses indicate the number of problem instances where no feasible solution was found. These instances were excluded from the calculation of average optimality gap.

may spend an excessive amount of time on one x value if there is even one particularly difficult mixed-integer subproblem. This also explains why several instances did not return a feasible solution; the first set of mixed-integer subproblems took excessively long to solve and did not finish before the time limit on the solution of (MP) was reached.

Conversely, the proposed modified integer L-shaped method displayed better scaling for this problem than in the first case study. The algorithm was able to solve all but one instance to optimality within the time limit. Additionally, the average computation time increased from 1,641 seconds for problems with 24 time periods and 8 scenarios to 10,449 seconds for problems with 48 time periods and 24 scenarios. This demonstrates the ability of the proposed algorithm to scale well in some problem instances that the standard integer L-shaped method is ill-suited to solve.

5 Conclusions

In this work, we have presented a modification to the integer L-shaped method with alternating cuts in which mixed-integer subproblems are terminated before the optimal solution is found and proven to be optimal. By doing this, our modified algorithm can efficiently generate no-good optimality cuts that are not supporting hyperplanes of the feasible region, which may be sufficient for fathoming the current node or otherwise advancing the solution of the master problem. This modification is inspired by the alternating cut generation strategy in the literature, which uses Benders cuts to serve the same purpose of efficiently separating the current master problem solution.

We performed a case study on two industrially relevant two-stage stochastic programs. In the first problem, an optimal set of modular, relocatable manufacturing units is sought for a supply chain network with uncertain future demands. In the second problem, an optimal design is sought for a renewables-based fuel and power production network with uncertain power generation profiles. The computational results of the studies suggest that the modified integer L-shaped method yields better scaling with the size and complexity of the mixed-integer subproblems. Additionally, it appears that the modification yields more consistent performance by avoiding cases where one

particularly difficult mixed-integer subproblem halts the progress of the algorithm and mitigates the advantage provided by parallelization.

It should be noted that while this work assumes relatively complete recourse, an extension of the proposed method to instances without relatively complete recourse is conceptually simple via the application of Benders feasibility cuts and no-good cuts. The reader is directed to [2] for details on incorporating these cuts. Additionally, while this work was written from the perspective of two-stage stochastic programming, the algorithm can be applied to any problem that has a suitable structure for the standard integer L-shaped method presented in [18].

Acknowledgement

This work was supported by the State of Minnesota through an appropriation from the Renewable Development Account to the University of Minnesota West Central Research and Outreach Center.

References

- [1] D. Weninger and L. Wolsey. *Benders' algorithm with (mixed)-integer subproblems*. Tech. rep. Center for Operations Research and Econometrics, 2019.
- [2] L. Wolsey. "Integer Programming". In: 2nd ed. Hoboken: John Wiley & Sons, Ltd, 2020, pp. 235–249. DOI: <https://doi.org/10.1002/9781119606475.ch12>.
- [3] A. Fakhri, M. Ghatee, A. Fragkogios, and G. K. Saharidis. "Benders decomposition with integer subproblem". In: *Expert Syst. Appl.* 89 (Dec. 2017), pp. 20–30. DOI: [10.1016/j.eswa.2017.07.017](https://doi.org/10.1016/j.eswa.2017.07.017).
- [4] H. D. Sherali and B. M. Fraticelli. "A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse". In: *J. Glob. Optim.* 22.1-4 (2002), pp. 319–342. DOI: [10.1023/a:1013827731218](https://doi.org/10.1023/a:1013827731218).
- [5] C. C. Carøe and J. Tind. "L-shaped decomposition of two-stage stochastic programs with integer recourse". In: *Math. Program.* 83.1-3 (Jan. 1998), pp. 451–464. DOI: [10.1007/BF02680570](https://doi.org/10.1007/BF02680570).
- [6] S. Sen and J. L. Higle. "The C3 Theorem and a D2 Algorithm for Large Scale Stochastic Mixed-Integer Programming: Set Convexification". In: *Math. Program.* 104.1 (Sept. 2005), pp. 1–20. DOI: [10.1007/s10107-004-0566-z](https://doi.org/10.1007/s10107-004-0566-z).
- [7] S. Sen and H. D. Sherali. "Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming". In: *Math. Program.* 106.2 (Apr. 2006), pp. 203–223. DOI: [10.1007/s10107-005-0592-5](https://doi.org/10.1007/s10107-005-0592-5).
- [8] R. Schultz, L. Stougie, and M. H. van der Vlerk. "Solving stochastic programs with integer recourse by enumeration: A framework using Gröbner basis". In: *Math. Program.* 83.1-3 (Jan. 1998), pp. 229–252. DOI: [10.1007/BF02680560](https://doi.org/10.1007/BF02680560).
- [9] A. Allman and Q. Zhang. "Branch-and-price for a class of nonconvex mixed-integer nonlinear programs". In: *J. Glob. Optim.* 81.4 (Dec. 2021), pp. 861–880. DOI: [10.1007/s10898-021-01027-w](https://doi.org/10.1007/s10898-021-01027-w).

- [10] T. Rathi, B. P. Riley, A. Flores-Quiroz, and Q. Zhang. "Column generation for multistage stochastic mixed-integer nonlinear programs with discrete state variables". In: *J. Glob. Optim.* (Mar. 2025).
- [11] J. Zou, S. Ahmed, and X. A. Sun. "Stochastic dual dynamic integer programming". In: *Math. Program.* 175.1-2 (May 2019), pp. 461–502. DOI: [10.1007/s10107-018-1249-5](https://doi.org/10.1007/s10107-018-1249-5).
- [12] G. Laporte and F. V. Louveaux. "The integer L-shaped method for stochastic integer programs with complete recourse". In: *Oper. Res. Lett.* 13.3 (Apr. 1993), pp. 133–142. DOI: [10.1016/0167-6377\(93\)90002-X](https://doi.org/10.1016/0167-6377(93)90002-X).
- [13] C. Li and I. E. Grossmann. "An improved L-shaped method for two-stage convex 0–1 mixed integer nonlinear stochastic programs". In: *Comput. Chem. Eng.* 112 (2018), pp. 165–179. DOI: [10.1016/j.compchemeng.2018.01.017](https://doi.org/10.1016/j.compchemeng.2018.01.017).
- [14] E. Sanci and M. S. Daskin. "An integer L-shaped algorithm for the integrated location and network restoration problem in disaster relief". In: *Transp. Res. Part B Methodol.* 145 (Mar. 2021), pp. 152–184. DOI: [10.1016/j.trb.2021.01.005](https://doi.org/10.1016/j.trb.2021.01.005).
- [15] B. Biesinger, B. Hu, and G. Raidl. "An Integer L-shaped Method for the Generalized Vehicle Routing Problem with Stochastic Demands". In: *Electron. Notes Discret. Math.* 52 (2016), pp. 245–252. DOI: [10.1016/j.endm.2016.03.033](https://doi.org/10.1016/j.endm.2016.03.033).
- [16] G. Laporte and F. V. Louveaux. "Solving Stochastic Routing Problems with the Integer L-Shaped Method". In: *Fleet Manag. Logist.* 1995. Boston, MA: Springer US, 1998, pp. 159–167. DOI: [10.1007/978-1-4615-5755-5_7](https://doi.org/10.1007/978-1-4615-5755-5_7).
- [17] Y. N. Hoogendoorn and R. Spliet. "An Improved Integer L-Shaped Method for the Vehicle Routing Problem with Stochastic Demands". In: *INFORMS J. Comput.* 35.2 (2023), pp. 423–439. DOI: [10.1287/ijoc.2023.1271](https://doi.org/10.1287/ijoc.2023.1271).
- [18] G. Angulo, S. Ahmed, and S. S. Dey. "Improving the Integer L-Shaped Method". In: *INFORMS J. Comput.* 28.3 (July 2016), pp. 483–499. DOI: [10.1287/ijoc.2016.0695](https://doi.org/10.1287/ijoc.2016.0695).
- [19] A. Allman and Q. Zhang. "Dynamic location of modular manufacturing facilities with relocation of individual modules". In: *Eur. J. Oper. Res.* 286.2 (Oct. 2020), pp. 494–507. DOI: [10.1016/J.EJOR.2020.03.045](https://doi.org/10.1016/J.EJOR.2020.03.045).
- [20] Gurobi Optimization LLC. *Gurobi Optimizer Reference Manual*. 2021.
- [21] I. Dunning, J. Huchette, and M. Lubin. "JuMP: A Modeling Language for Mathematical Optimization". In: *SIAM Rev.* 59.2 (Jan. 2017), pp. 295–320. DOI: [10.1137/15M1020575](https://doi.org/10.1137/15M1020575).
- [22] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Rev.* 59.1 (Jan. 2017), pp. 65–98. DOI: [10.1137/141000671](https://doi.org/10.1137/141000671).
- [23] Q. Zhang, M. Martín, and I. E. Grossmann. "Integrated design and operation of renewables-based fuels and power production networks". In: *Comput. Chem. Eng.* 122 (Mar. 2019), pp. 80–92. DOI: [10.1016/J.COMPCHEMENG.2018.06.018](https://doi.org/10.1016/J.COMPCHEMENG.2018.06.018).

6 Appendix: Model for Case Study 2

Nomenclature

Indices / sets

$i \in \mathcal{D}_j$	set of processes that consume resource j
$h \in H$	seasons
$i \in I$	processes
$j \in J$	resources
$l \in L$	segments in piecewise-linear approximations
$m \in M$	operating modes
$t \in T$	time periods, $T = \{-\theta^{\max} + 1, -\theta^{\max} + 2, \dots, 0, 1, \dots, T \}$
$i \in \mathcal{U}_j$	set of processes that produce resource j

Subsets

$\bar{J}$	resources with demand
$\hat{J}$	resources that must not be discharged
L_i	segments in piecewise-linear approximation for process i
M_i	operating modes in process i
SQ_i	predefined sequences of mode transitions in process i
$\bar{T}_h$	time periods in season h , $\bar{T}_h = \{1, 2, \dots, \bar{T}_h \}$
TR_i	possible mode transitions in process i
$\overline{TR}_{im}$	modes from which mode m can be directly reached in process i
$\widehat{TR}_{im}$	modes which can be directly reached from mode m in process i

Parameters

$B_{jht}^{\max}$	maximum amount of resource j that can be consumed by the process network in time period t of season h
$C_i^{\max}$	maximum production capacity for process i
$\bar{C}_j^{\max}$	maximum storage capacity for resource j
$\hat{C}_{il}$	production capacity for process i at right end point of segment l
$\hat{C}_{im}^{\max}$	maximum production amount in mode m of process i
$\hat{C}_{im}^{\min}$	minimum production amount in mode m of process i
D_{jht}	demand for resource j in time period t of season h
$\bar{M}$	big-M parameter in rate-of-change constraint
n_h	number of times the representative scheduling horizon of season h is repeated
α_j	fixed capital cost for storing resource j
δ_{imh}	fixed cost for operating in mode m of process i in season h
Δt	length of one time period
$\bar{\Delta}_{im}^{\max}$	maximum rate of change
γ_{imh}	unit cost for operating in mode m of process i in season h
ϵ_{jh}	fractional loss from storing resource j in season h

η_{ihts}	fractional availability of production capacity in process i in time period t of season h
	mode m of process i in season h
$\theta_{imm'}$	minimum stay time in mode m' of process i after switching from mode m to m'
$\bar{\theta}_{imm'm''}$	fixed stay time in mode m' of the predefined sequence (m, m', m'') in process i
$\theta^{\max}$	maximum minimum or predefined stay time in a mode
ρ_{imjh}	conversion factor for resource j with respect to the reference resource in
ϕ_{jh}	unit cost for purchasing resource j in season h
ψ_{jh}	unit cost for discharging resource j in season h

Unrestricted continuous variables

$\bar{Q}_{jhs}$	excess inventory for resource j in season h
-----------------	---

Nonnegative continuous variables

B_{jhts}	amount of resource j consumed by the process network in time period t of season h
C_i	production capacity for process i
$\tilde{D}_{jhts}$	amount of demand of resource j not met in season h and time t
P_{ihts}	amount of reference resource produced by process i in time period t of season h
$\bar{P}_{imhts}$	amount of reference resource produced in mode m of process i in time period t of season h
Q_{jhts}	inventory level for resource j at time period t of season h
S_{jhts}	amount of resource j discharged from the network in time period t of season h

Binary variables

x_i	equals 1 if process i is selected
y_{imhts}	equals 1 if process i operates in mode m in time period t of season h
$z_{imm'hts}$	equals 1 if operation of process i switched from mode m to mode m' at time t of season h

Problem Formulation

$$\begin{aligned} \text{minimize} \quad & \sum_i \sigma_i C_i^{\text{nameplate}} x_i + \sum_s p_s \sum_h \sum_{t \in \bar{T}_h} n_h \left[\sum_i \sum_{m \in M_i} (\delta_{imh} y_{imhts} + \gamma_{imh} \bar{P}_{imhts}) \right. \\ & \left. + \sum_j \phi_{jh} B_{jhst} + \sum_j \psi_{jh} S_{jhst} \right] \end{aligned} \quad (25)$$

$$\text{subject to} \quad C_i = C_i^{\text{nameplate}} x_i \quad \forall i \quad (26)$$

$$x_i \in \{0, 1\} \quad \forall i \quad (27)$$

$$x_i \leq \sum_{i' \in \mathcal{U}_j} x_{i'} \quad \forall i \in \mathcal{D}_j \quad (28)$$

$$\sum_{m \in M_i} y_{imhts} = x_i \quad \forall i, h, t \in \bar{T}_h, s \quad (29)$$

$$P_{ihts} = \sum_{m \in M_i} \bar{P}_{imhts} \quad \forall i, h, t \in \bar{T}_h, s \quad (30)$$

$$\tilde{C}_{im}^{\min} y_{imhts} \leq \bar{P}_{imhts} \leq \tilde{C}_{im}^{\max} y_{imhts} \quad \forall i, m \in M_i, h, t \in \bar{T}_h, s \quad (31)$$

$$y_{imhts} \in \{0, 1\} \quad \forall i, m \in M_i, h, t \in \bar{T}_h, s \quad (32)$$

$$\begin{aligned} -\bar{\Delta}_{im}^{\max} - \bar{M}(2 - y_{imhts} - y_{imh,t-1,s}) &\leq \bar{P}_{imhts} - \bar{P}_{imh,t-1,s} \\ &\leq \bar{\Delta}_{im}^{\max} + \bar{M}(2 - y_{imhts} - y_{imh,t-1,s}) \quad \forall i, m \in M_i, h, t \in \bar{T}_h, s \end{aligned} \quad (33)$$

$$\sum_{m' \in \overline{TR}_{im}} z_{im'mh,t-1,s} - \sum_{m' \in \widehat{TR}_{im}} z_{imm'h,t-1,s} = y_{imhts} - y_{imh,t-1,s} \quad \forall i, m, h, t, s \quad (34)$$

$$z_{imm'hts} \in \{0, 1\} \quad \forall i, (m, m') \in TR_i, h, t \in \bar{T}_h, s \quad (35)$$

$$y_{im'hts} \geq \sum_{k=1}^{\theta_{imm'}} z_{imm'h,t-k,s} \quad \forall i, (m, m') \in TR_i, h, t \in \bar{T}_h, s \quad (36)$$

$$z_{imm'h,t-\bar{\theta}_{imm'm''},s} = z_{im'm''hts} \quad \forall i, (m, m', m'') \in SQ_i, h, t \in \bar{T}_h, s \quad (37)$$

$$Q_{jhst} = (1 - \epsilon_{jh}) Q_{jh,t-1,s} + \sum_i \sum_{m \in M_i} \rho_{imjh} \bar{P}_{imhts} + B_{jhst} - S_{jhst} \quad \forall j, h, t, s \quad (38)$$

$$P_{ihts} \leq \eta_{ihts} C_i^{\text{nameplate}} x_i \quad \forall i, h, t \in \bar{T}_h, s \quad (39)$$

$$Q_{jhst} \leq \bar{C}_j \quad \forall j, h, t \in \bar{T}_h, s \quad (40)$$

$$B_{jhst} \leq B_{jht}^{\max} \quad \forall j, h, t \in \bar{T}_h, s \quad (41)$$

$$S_{jhst} + \tilde{D}_{jhst} \geq D_{jht} \quad \forall j \in \bar{J}, h, t \in \bar{T}_h, s \quad (42)$$

$$S_{jhst} = 0 \quad \forall j \in \hat{J}, h, t \in \bar{T}_h, s \quad (43)$$

$$y_{imh,0,s} = y_{imh,|\bar{T}_h|,s} \quad \forall i, m \in M_i, h, s \quad (44)$$

$$z_{imm'hts} = z_{imm'h,t+|\bar{T}_h|,s} \quad \forall i, (m, m') \in TR_i, h, -\theta_i^{\max} + 1 \leq t \leq -1, s \quad (45)$$

$$y_{imh,|\bar{T}_h|,s} = y_{im,h+1,0,s} \quad \forall i, m \in M_i, h \in H \setminus \{|H|\}, s \quad (46)$$

$$z_{imm'h,t+|\bar{T}_h|,s} = z_{imm',h+1,t,s} \quad \forall I, m, m', h \in H \setminus \{|H|\}, -\theta_i^{\max} + 1 \leq t \leq -1, s \quad (47)$$

$$\bar{Q}_{jhst} = Q_{jh,|\bar{T}_h|,s} - Q_{jh,0,s} \quad \forall j, h, s \quad (48)$$

$$Q_{jh,0,s} + n_h \bar{Q}_{jhs} = Q_{j,h+1,0,s} \quad \forall j, h \in H \setminus \{|H|\}, s \quad (49)$$

$$Q_{j,|H|,0,s} + n_{|H|} \bar{Q}_{j,|H|,s} \geq Q_{j,1,0,s} \quad \forall j, s \quad (50)$$

Here, (26) are constraints on the investment decisions, x_i , that relate the investment decision to the capacity of the process units. Constraints (28) are constraints added to the model in this work. They encode the logic that processes should not be built if there is no way to produce an intermediate resource that they consume ($\mathcal{D}_j$ are processes that consume resource j , and $\mathcal{U}_j$ are processes that produce j). These constraints are added to improve the formulation of the master problem without changing the optimal solution. Constraints (29) make it so that a process must be in an operating mode if it is built, and cannot operate if it is not built. Constraints (30) model the production of resources to the mode-specific production rate of each process. Constraints (31) apply bounds on the mode-specific production rate and ensure that a mode can only produce a resource if the process is operating in that mode. Constraints (33) are big-M constraints that apply ramping constraints to the mode-specific production rates. Constraints (34) make $z_{imm'hts}$ encode whether or not process i transitions from mode m to mode m' at the end of time period t . Constraints (36) enforce a minimum amount of time that a process must remain in a mode after transitioning to it. Constraints (37) ensure that processes transition between modes at the right time if the modes are part of a fixed sequence of transitions. Constraints (38) are mass balances. Constraints (39)–(43) apply bounds to the various components of the mass balances. Constraints (44) and (45) enforce the cyclic operation of the system over the courses of each season, h . Constraints (46) and (47) relate the system state at the end of one season to the initial state of the next season. Constraints (48)–(50) model the accumulation of resources over the course of a cyclic season. The objective function includes components for capital costs, mode-specific operating costs, mode-specific production costs, raw materials costs, and the costs of discharging certain dischargeable resources. Readers are directed to [23] for further discussion of the model components.