Indicator Cuts for Benders Decomposition with Mixed-Integer Subproblems

Bingqian Liu

Centre Inria de l'Université de Bordeaux
F-33405 Talence, France
bingqian.liu@math.u-bordeaux.fr

Céline Gicquel

LISN, Université Paris-Saclay / CNRS
Gif-sur-Yvette, France
gicquel@lisn.fr

Preprint, August 2026

Abstract

Classical Benders decomposition fails when the subproblem is a mixed-integer program, due to the absence of strong duality. We propose a novel class of dual-free indicator cuts that are applicable to all Benders-decomposable problems with a pure-integer master problem and mixed-integer linear programming (MILP) subproblems. These cuts are derived from the monotonicity property of the subproblem value function, allowing us to construct valid cuts without relying on duality. We also introduce a theoretical framework for comparing the effectiveness of different indicator cuts. This framework unifies several previously disconnected methods that exploit subproblem monotonicity. Two MILP formulations of the proposed cuts are developed, along with two implementation techniques to improve computational performance. The proposed method is tested on a multi-phase energy system design problem. Numerical results show that the indicator cuts significantly outperform CPLEX in both runtime and solution quality, and the empirical performance aligns with the theoretical predictions.

1 Introduction

One of the main challenges of large-scale optimization problems is that the difficulty of solving them tends to increase exponentially with the size of the

problem. A widely used approach to address this challenge without sacrificing optimality is to apply decomposition algorithms. Among them, Benders decomposition (BD), proposed in the 1960s by [Benders, 1962], has achieved great success.

Optimization problems suitable for BD exhibit a specific structure: the problem becomes tractable once a small set of variables is fixed. These variables are referred to as complicating variables. The original version of BD assumes that the problem decomposes into a set of linear programming (LP) subproblems when the complicating variables are fixed. A sequence of Benders cuts is then iteratively generated based on the dual solutions of the subproblems. These cuts form an outer approximation of the objective function and ensure that the algorithm eventually converges to the optimal solution. A generalized version of BD was proposed in [Geoffrion, 1972], which allows the subproblems to be nonlinear programs. However, classical BD still requires the subproblems to be convex in order to ensure strong duality, so that the cuts can be derived from their dual problems.

When the subproblems involve integer variables, classical BD is no longer valid due to the lack of strong duality. A variety of alternative approaches have been developed to address this issue. However, these approaches often suffer from limitations such as the reliance on binary master variables, the dependence on auxiliary problem solutions or the need to exploit problem-specific features. In this article, we propose a generic approach for solving a wide range of Benders-decomposable mixed-integer linear programming (MILP) problems. More precisely, we assume that the master problem is a pure integer program, while the subproblems can be general MILPs. The Benders cuts, referred to as *indicator cuts* in this context, are constructed based on the partial order property of the subproblems' value function. Our approach exploits the fact that the optimal objective value of each lower-level subproblem exhibits a monotonic relationship with respect to variations in the upper-level variables.

The main contributions of this work are as follows:

- We propose a new type of Benders cut, called the *indicator Benders cut*, based on the monotonicity of the subproblem value function. These cuts are applicable to all Benders-decomposable problems in which the master problem is a pure integer program and the subproblems are MILPs.

- We develop a theoretical framework for analyzing the effectiveness of indicator Benders cuts. This framework provides a basis for selecting and comparing different cuts that all guarantee finite and optimal convergence of the iterative algorithm. It also unifies several previously independent works that exploit subproblem monotonicity, such as the L-shaped method [Laporte and Louveaux, 1993], combinatorial cuts [Codato and Fischetti, 2006], monotonic logic-based Benders cuts [Forbes et al., 2024], MIDAS [Philpott et al., 2020], and generalized Benders cuts using indicator functions [Liu et al., 2024].

- We propose two formulations for the indicator Benders cuts, which are non-continuous and non-convex. In addition, we introduce two algorith-

mic enhancement techniques to accelerate the convergence of the iterative solution process.

The remainder of the article is organized as follows. Section 2 reviews existing methods for Benders decomposition with mixed-integer subproblems. Section 3 introduces the proposed indicator Benders cuts, which are formulated using indicator functions and guarantee the convergence of the algorithm. Section 4 presents two MILP formulations of these cuts. Section 5 introduces two enhancement techniques for improving their computational performance. Finally, Section 6 reports numerical results on a multi-phase design problem for a district cooling system.

2 Literature Review

The original BD algorithm was proposed by [Benders, 1962]. The classical BD algorithm leverages the dual solutions of the subproblems to generate Benders cuts. A generalized version of BD was subsequently proposed by [Geoffrion, 1972], requiring the subproblems be convex programs. In both cases, the strong duality property of the subproblems is essential to ensure the optimal convergence of the algorithm.

When the subproblems involve integer variables, they may exhibit a positive duality gap. In such cases, the classical Benders-decomposition framework, which relies on strong duality, may become inapplicable and can fail to converge. However, Benders cuts do not have to be derived solely from the dual of the subproblems. As summarized by [Bolusani and Ralphs, 2022], generalized Benders cuts can be employed to ensure the finite convergence of the algorithm even when the subproblems contain integer variables. These generalized cuts can be non-linear or even non-convex, as long as they are strong enough to guide convergence.

In this section, we review several techniques for handling mixed-integer subproblems within the BD framework, following the classification proposed by [Bolusani and Ralphs, 2022].

2.1 No-good Cut

When the complicating variables are binary, no-good cuts can be used to ensure the finite convergence of the BD algorithm [Wolsey, 2020]. The concept of optimality cuts in this context was first introduced by [Laporte and Louveaux, 1993], who applied an integer L-shaped method to tackle integer subproblems with binary complicating variables. Later, feasibility cuts in the form of combinatorial cuts were proposed by [Codato and Fischetti, 2006]. When the complicating variables are integer, a common approach is to apply binary expansion, replacing each upper-level integer variable with a set of binary variables [Owen and Mehrotra, 2002]. Although this transformation enables the use of no-good cuts, it also increases the problem size significantly.

The main drawback of no-good cuts is that, at each iteration, they only either improve the lower bound for a single feasible solution or exclude a single infeasible solution from the master problem feasible space [Angulo et al., 2016], which may lead to slow convergence. To improve efficiency, some researchers propose generating classical linear Benders cuts from the linear relaxation of the subproblems and combining them with no-good cuts [Papadakos, 2008, Liu et al., 2024].

2.2 Logic-based Benders Cut

Logic-based Benders decomposition (LBBD), introduced by [Hooker and Ottosson, 2003] and [Hooker, 2011], extends the classical BD framework by generating cuts based on logical inference rather than duality. In LBBD, the original problem is decomposed into a master problem and one or more subproblems. However, unlike classical BD, which is restricted to linear or convex subproblems, LBBD allows the subproblem to be an arbitrary optimization problem, such as integer programming [Forbes et al., 2024], constraint programming [Hooker, 2005], or feasibility checking [Martínez et al., 2022]. The Benders cuts are derived from the logical structure of the subproblems, exploiting problem-specific inference rather than relying solely on dual variables. This approach has proven effective in various domains, including task scheduling [Hooker, 2007], facility location [Fazel-Zarandi and Beck, 2012], and network design [Gendron et al., 2014].

However, LBBD typically requires a deep understanding of the problem's logical structure to formulate effective, problem-specific cuts [Hooker, 2019]. As a result, the method may be less accessible to practitioners without specialized expertise.

2.3 Lipschitz Cut and Scaled Cut

The Lipschitz cut was introduced by [Ahmed et al., 2022] as a form of generalized Benders cut. It consists of the sum of a linear cut and a reversed norm term. The reversed norm term is defined as the L^1 distance between a candidate solution and the current solution, multiplied by a negative coefficient. Since this reversed norm is a continuous function, the value function is assumed to be continuous, meaning that every integer-feasible solution in the master problem leads to a feasible subproblem. [van der Laan and Romeijnders, 2023] generalized this approach. They referred to the original Lipschitz cut from [Ahmed et al., 2022] as an "unscaled cut," and proposed a method for generating a scaled cut by solving a sequence of optimization problems. The resulting scaled cut is linear and added to the master problem. The authors also proved the finite convergence of the algorithm using this approach.

2.4 Lagrangian Dual

Lagrangian dual cuts represent another class of convex cuts that can improve the formulation of the master problem [Chen and Luedtke, 2022, Li and Grossmann, 2018,

Rahmaniani et al., 2020]. To construct such cuts, local copies of the upper-level variables are introduced, along with constraints enforcing equality between these copies and the original variables. A Lagrangian relaxation is then applied to these linking constraints, and the corresponding Lagrangian dual problem is solved to obtain optimal values for the Lagrangian multipliers.

When the master problem is a pure 0–1 linear program, these cuts can ensure finite and optimal convergence of the Benders decomposition algorithm [Zou et al., 2019]. However, this property does not hold when the master problem includes general integer or continuous variables [van der Laan and Romeijnders, 2023], as Lagrangian cuts may not be tight in such cases. One way to address this issue is to perform a binary expansion of the upper-level variables [Füllner and Rebennack, 2022], but this can result in a significant loss of computational efficiency [Owen and Mehrotra, 2002].

Furthermore, generating Lagrangian dual cuts typically involves solving the Lagrangian dual problem repeatedly, which can be computationally expensive [Chen and Luedtke, 2022].

2.5 Monotonicity of the subproblem value function

Exploiting the monotonicity of the subproblem value function is another effective approach to generate generalized Benders cuts. [Philpott et al., 2020] proposed using step functions to approximate a monotonic Bellman function. [Forbes et al., 2024] studied a case where the master problem contains only integer variables. They assumed that the subproblem value function is non-increasing and constructed a monotonic cut inspired by the conventional no-good cut.

Although not a Benders decomposition method, [Yıldız et al., 2022] is closely related to this line of work. The authors exploited the monotonicity of block-level value functions with respect to the right-hand-side parameters of linking constraints, and used this information to generate branching constraints within a decomposition branching framework. They also introduced a non-negative constraint aggregation technique to reduce the number of resulting branching constraints.

In another related work, [Liu et al., 2024] formulated Benders cuts using an indicator function for the specific case where the coefficient matrix of upper-level variables in the coupling constraints has non-negative entries. In their setting, the subproblem value function has a partial-order structure. Once the operational subproblem has been solved for a given upper-level solution, its optimal value provides a valid lower bound for the subproblem value associated with any other comparable upper-level solution that is componentwise smaller.

2.6 Parametric cutting plane

Several researchers have explored enhancing the continuous relaxation of the subproblems rather than solving them directly as mixed-integer programs. One line of work proposes the use of parametric cutting planes to tighten the continuous relaxation of second-stage problems before generating classical Benders cuts

based on the dual solution of the subproblems [Gade et al., 2014, Ntaimo, 2010, Ntaimo, 2013, Qi and Sen, 2017, Sen and Higle, 2005, Zhang and Küçükyavuz, 2014]. The cuts derived from the solutions of these strengthened subproblems can significantly improve the convergence of the algorithm. However, these methods often rely on the assumption that the master problem is a pure binary program in order to guarantee finite convergence.

2.7 Cuts from Leaf Nodes of the Branch-and-Bound Tree

Another approach exploits the branch-and-bound tree generated when solving the MILP subproblems. Classical Benders cuts can be derived from the dual solutions of the LP relaxations at terminal nodes, and a strong cut can be obtained by taking their pointwise infimum [Wolsey, 1981, Bolusani and Ralphs, 2022]. In a related approach, [Sen and Sherali, 2006] use terminal-node information from a branch-and-cut tree to build a disjunctive approximation of the subproblem value function and derive valid cuts for the master problem.

2.8 Summary and research gap

Overall, the existing literature provides several alternatives to classical Benders decomposition when the subproblems contain integer variables. However, these approaches often rely on binary master variables, problem-specific logical inference, auxiliary optimization problems, or a particular problem structure. In particular, existing methods exploiting subproblem monotonicity remain largely problem-specific and do not provide a unified framework for characterizing, comparing, and leveraging such monotonicity in general Benders-decomposable problems.

This work addresses this gap by developing indicator Benders cuts for problems with a pure-integer master problem and MILP subproblems. It also provides a theoretical framework to compare monotonicity-based cuts, relate existing methods within a common perspective, and study the trade-off between cut strength and formulation complexity.

3 Indicator Benders cut: theory

3.1 Generalized Benders decomposition framework

A typical optimization problem to which generalized Benders decomposition can be applied is formulated as follows:

$$\begin{aligned} \text{minimize} \quad & c_1 x + c_2 y && (1)\\ \text{subject to} \quad & By \leq Ax + d && (2)\\ & x \in \mathcal{X} && (3)\\ & y \in \mathcal{Y} && (4) \end{aligned}$$

In this model, x and y are decision variables. The sets $\mathcal{X}$ and $\mathcal{Y}$ represent the feasible regions for x and y, respectively. In this paper, we assume that $\mathcal{X} \subseteq \mathbb{Z}^m$ is bounded, and $\mathcal{Y} \subseteq \mathbb{Z}^p \times \mathbb{R}^{n-p}$, meaning that x consists of integer variables, while y includes both integer and continuous variables. The matrix $A \in \mathbb{R}^{k \times m}$, matrix $B \in \mathbb{R}^{k \times n}$, and vector $d \in \mathbb{R}^k$ are parameters of the problem. We refer to x as the upper-level variables, to y as the lower-level variables and to $By \leq Ax + d$ as the coupling constraints.

The problem can be decomposed into a master problem (MP) and a subproblem (SP).

$$\begin{array}{lll} \textbf{MP} & \text{minimize} & c_1 x + Z(x) \\ & \text{subject to} & x \in \mathcal{X} \end{array}$$

$$\begin{array}{llll} \textbf{SP}(x) & Z(x) := & \text{minimize} & c_2 y \\ & & \text{subject to} & By \leq Ax + d \\ & & & y \in \mathcal{Y} \end{array}$$

Problem MP involves only the upper-level variables. Problem SP(x) determines the optimal value of the lower-level variables y as a function of the upper-level variable value x. Let $Z(\cdot)$ denote its value function. We assume that the value function $Z(x)$ is bounded and we adopt the notation that $Z(x) = +\infty$ if $\mathbf{SP}(x)$ is infeasible.

In general, there is no analytic closed-form expression known for function $Z(\cdot)$. As noted by [Bolusani and Ralphs, 2022], even if an explicit form exists in theory, its description is most often of exponential size and would thus be numerically intractable. One way to overcome this difficulty consists in using a generalized Benders decomposition algorithm to iteratively build an outer approximation of $Z(\cdot)$.

At iteration $i \geq 1$ of such an algorithm, the master problem is reformulated as:

$$\begin{array}{lll} \mathbf{MP}^i & \text{minimize} & c_1 x + \eta \qquad (5) \\ & \text{subject to} & x \in \mathcal{X} \qquad (6) \\ & & \eta \geq f(x, \hat{x}^j) \qquad j \in \{0, \cdots, i-1\} \qquad (7) \end{array}$$

Problem $\mathbf{MP}^i$ thus represents the master problem to be solved at iteration $i \geq 1$ of the algorithm, using the outer approximation of $Z(\cdot)$ constructed from previous iterations. Here, $f(\cdot, \hat{x}^j)$ is a function of x parameterized by $\hat{x}^j$, the optimal solution of the master problem $\mathbf{MP}^j$ solved at iteration j. Constraints (7) are referred to as generalized Benders cuts. The cut generated at iteration j can be classified as either an optimality cut (in case $\mathbf{SP}(\hat{x}^j)$ is feasible) or a feasibility cut (in case $\mathbf{SP}(\hat{x}^j)$ is infeasible). Note that at the beginning of iteration $i = 0$, $\mathbf{MP}^0$ does not involve any generalized Benders cut and is thus formulated as Problem (5)-(6).

The choice of function f is crucial for ensuring the finite convergence of the generalized BD algorithm. As noted by [Bolusani and Ralphs, 2022], a sufficient condition for the finite and optimal convergence of the algorithm is that, for all $j \in \{0, \ldots, i-1\}$, the function $f(\cdot, \hat{x}^j)$ satisfies the following two criteria:

1. $f(\cdot, \hat{x}^j)$ is a general dual function of $Z(\cdot)$, i.e., $f(x, \hat{x}^j) \leq Z(x)\ \forall x \in \mathcal{X}$;
2. $f(\cdot, \hat{x}^j)$ is strong at point $\hat{x}^j$, i.e., $f(\hat{x}^j, \hat{x}^j) = Z(\hat{x}^j)$.

If the set $\mathcal{Y}$ is convex, then by duality theory, $Z(\cdot)$ is a convex function, allowing the application of classical BD. In this case, $f(\cdot, \hat{x}^j)$ is a linear function, whose coefficients are determined by $\hat{x}^j$ and by the solution of the dual of $\mathbf{SP}(\hat{x}^j)$. However, when **SP** includes integer variables, $Z(\cdot)$ may become non-convex. In what follows, we investigate how to build an outer approximation of function $Z(\cdot)$ in this case by using non-convex strong dual functions f.

3.2 Partial order property and indicator Benders cut

Subproblem $\mathbf{SP}(x)$ is a parametric MILP, in which the right-hand side of the coupling constraints is parameterized by $Ax + d$. Since all coupling constraints are expressed as less-than-or-equal-to inequalities, problem **SP** exhibits a partial ordering property with respect to different values of x.

Lemma 3.1. *For any pair $(x^1, x^2) \in \mathcal{X}^2$ of upper-level solutions, we have:*

$$Ax^1 \geq Ax^2 \implies Z(x^1) \leq Z(x^2) \tag{8}$$

Proof. Since $Ax^1 \geq Ax^2$, it follows that $\{y | By \leq Ax^2 + d\} \subseteq \{y | By \leq Ax^1 + d\}$. Therefore, the feasible region of $\mathbf{SP}(x^2)$ is included in that of $\mathbf{SP}(x^1)$. As $\mathbf{SP}(x^1)$ and $\mathbf{SP}(x^2)$ have the same objective coefficients, we have $Z(x^1) \leq Z(x^2)$. □

Note that Lemma 3.1 also provides information regarding the feasibility of the subproblems: if x^2 is feasible, then x^1 is feasible. Conversely, if x^1 is infeasible, then x^2 is also infeasible.

Leveraging the partial order property, for each value $\hat{x} \in \mathcal{X}$ and its corresponding subproblem objective value $Z(\hat{x})$, we define a strong general dual function for $Z(x)$ using the following indicator function:

$$\mathbb{1}_{Ax \leq A\hat{x}} = \begin{cases} 1 & \text{if } Ax \leq A\hat{x} \\ 0 & \text{else} \end{cases} \tag{9}$$

Proposition 3.1. *For any $\hat{x} \in \mathcal{X}$, the function f defined by*

$$f(x, \hat{x}) = (Z(\hat{x}) + M)\mathbb{1}_{Ax \leq A\hat{x}} - M, \quad \forall x \in \mathcal{X} \tag{10}$$

is a strong dual function of the value function $Z(\cdot)$. In the function, M is a number large enough that $-M$ is a lower bound of function Z.

Proof. For any $x \in \mathcal{X}$ such that $Ax \leq A\hat{x}$, by Lemma 3.1, we obtain that $f(x, \hat{x}) = Z(\hat{x}) \leq Z(x)$. For any other value of $x \in \mathcal{X}$, by the definition of the indicator function, we have $f(x, \hat{x}) = -M \leq Z(x)$. Therefore, $f(x, \hat{x}) \leq Z(x)$ $\forall x \in \mathcal{X}$, implying that $f(x, \hat{x})$ is a dual function of $Z(x)$. Since $f(\hat{x}, \hat{x}) = Z(\hat{x})$, $f(x, \hat{x})$ is a strong dual function of $Z(x)$. □

The generalized Benders cut added to the master problem at the end of iteration i is given by:

$$\eta \geq (Z(\hat{x}^i) + M)\mathbb{1}_{Ax \leq A\hat{x}^i} - M \tag{11}$$

In theory, using such an indicator cut at each iteration to progressively strengthen the master problem formulation guarantees the finite and optimal convergence of the generalized Benders decomposition algorithm [Bolusani and Ralphs, 2022]. However, in practice, incorporating this cut into the master problem formulation can be computationally challenging, as it introduces a non-convex constraint.

A widely used approach to formulate such an indicator cut is through a disjunctive formulation, which expresses the non-convex cut as the union of a collection of polyhedral sets [Balas, 2018]. For general integer variables x, this method requires introducing $\mathcal{O}(k + 1)$ new auxiliary binary variables in the master problem formulation, where k denotes the number of rows of matrix A. Consequently, each iteration of the algorithm adds roughly $k+1$ auxiliary binary variables together with their associated constraints, substantially increasing the computational effort needed to solve the master problem.

3.3 Effective equivalent of the coupling matrix

To reduce the formulation burden of indicator cuts, we propose a method that reduces the number of binary variables and linear constraints introduced at each iteration. The main idea of this method is to leverage the fact that the set of constraints $Ax \leq A\hat{x}$ often contains redundant inequalities. Thus, we aim to represent the set $\{x \in \mathcal{X} | Ax \leq A\hat{x}\}$ more compactly as $\{x \in \mathcal{X} | A'x \leq A'\hat{x}\}$ where $A' \in \mathbb{R}^{l \times m}$ is a matrix involving a number of rows $l \ll k$, much smaller than the one of the original matrix A.

To construct such a matrix A', we study the mathematical properties of the polyhedral cone defined as the set of all vectors satisfying the system of homogeneous linear inequalities associated to matrix $A \in \mathbb{R}^{k \times m}$. Let $K(A) := \{x \in \mathbb{R}^m \mid Ax \leq 0\}$ denote this cone.

The condition $Ax \leq A\hat{x}$ can be rewritten as $x - \hat{x} \in K(A)$, showing that the formulation of the indicator Benders cut depends solely on the cone $K(A)$ and not on the particular matrix A that induces it. Hence, we may search for an alternative matrix $A' \in \mathbb{R}^{l \times m}$ such that $K(A') = K(A)$.

Definition 3.1 (Effective equivalent)**.** Let $A \in \mathbb{R}^{k \times m}$ and define $K(A) := \{x \in \mathbb{R}^m \mid Ax \leq 0\}$. A matrix $A' \in \mathbb{R}^{\ell \times m}$ is called an *effective equivalent* of A if $K(A') = K(A)$. In this case, we write $A' \sim_e A$.

Although the relation $Ax \leq A\hat{x}$ is sometimes referred to as a *generalized inequality* with respect to the cone $K(A)$, we avoid this terminology to maintain consistency with our formulation style.

Let us now consider the polar cone of $K(A)$, denoted by $K^o(A)$ and defined as:

$$K^o(A) := \{v \in \mathbb{R}^m \mid v^T x \leq 0, \forall x \in K(A)\}. \tag{12}$$

By cone duality, for two matrices A and A', if $K^o(A) = K^o(A')$, then $K(A) = K(A')$, and therefore $A \sim_e A'$. Our search for an effective equivalent matrix A' with fewer rows leverages this property and thus proceeds by an analysis of the properties of the polar cone $K^o(A)$.

Let $lin(A)$ and $cone(A)$ denote, respectively, the set of all linear and conic combinations of the rows of $A \in \mathbb{R}^{k \times m}$:

$$lin(A) := \{v \in \mathbb{R}^m \mid v = A^T\lambda,\ \lambda \in \mathbb{R}^k\}. \tag{13}$$

$$cone(A) := \{v \in \mathbb{R}^m \mid v = A^T\lambda,\ \lambda \in \mathbb{R}^k_+\}. \tag{14}$$

We show in what follows that $K^o(A)$ is equal to $cone(A)$.

Proposition 3.2. *The polar cone $K^o(A)$ is equal to the set of all conic combinations of the row vectors of matrix A, i.e.:*

$$K^o(A) = cone(A). \tag{15}$$

Proof. We first prove that $cone(A) \subseteq K^o(A)$. Let $v \in cone(A)$, so that $v = A^T\lambda$ for some $\lambda \in \mathbb{R}^k_+$. Then for all $x \in K(A)$,

$$v^T x = (A^T\lambda)x = \lambda^T(Ax) \leq 0, \qquad \forall x \in K(A) \tag{16}$$

as $\lambda \geq 0$ and $Ax \leq 0$ by definition of $K(A)$. This shows that $v \in K^o(A)$ and that $cone(A) \subseteq K^o(A)$.

We now prove that $K^o(A) \subseteq cone(A)$. Let $v \in K^o(A)$ and consider the following linear program:

$$\begin{aligned} maximize \quad & v^T x && \text{(17a)} \\ & Ax \leq 0 && \text{(17b)} \\ & x \in \mathbb{R}^m && \text{(17c)} \end{aligned}$$

Since $v \in K^o(A)$, the objective satisfies $max\{v^T x, x \in K(A)\} \leq 0$. Furthermore, $x = 0$ is feasible, so the optimal value is 0, i.e., $max\{v^T x, x \in K(A)\} = 0$. Problem (17a)-(17c) is thus a bounded linear program with an optimal value equal to 0. The corresponding dual problem is given by:

$$\begin{aligned} \text{minimize} \quad & 0 \\ \text{subject to} \quad & A^T\mu = v \\ & \mu \in \mathbb{R}^k_+ \end{aligned}$$

As Problem (17a)-(17c) is bounded, its dual problem is feasible. Consequently, there exists a vector $\mu \in \mathbb{R}^k_+$ such that $A^T\mu = v$. In other words, it is possible to express v as a conic combination of row vectors of A. This shows that $v \in cone(A)$ and that $K^o(A) \subseteq cone(A)$.
Finally, we conclude that $K^o(A) = cone(A)$. □

Property 3.2 shows that the search for an effective equivalent matrix A' with fewer rows depends on the geometric structure of $K^o(A)$. In particular, one must determine whether $K^o(A)$ is pointed, i.e., whether the origin is an extreme point of the cone. In case $K^o(A)$ is pointed, it can be finitely generated from a unique set of extreme rays. This set can be identified among the set of vectors generating $K^o(A)$, i.e. among the set of rows of matrix A since $K^o(A) = cone(A)$. Consequently, matrix A' can be constructed by selecting precisely the rows of A corresponding to extreme rays of $K^o(A)$. In case $K^o(A)$ is not pointed, it contains a lineality space and its structure cannot be captured solely by its rows. Therefore, constructing A' is no longer straightforward.

In Section 3.4, we discuss how to construct an effective equivalent matrix A' in case $K^o(A)$ is pointed. Section 3.5 addresses the general case. Finally, in Section 3.6, we introduce the notion of cut effectiveness, allowing for less effective but computationally simpler formulations, and compare different indicator cuts accordingly.

3.4 Reduction of coupling constraints: pointed $K^o(A)$

In this section, we assume that the polar cone $K^o(A)$ is pointed. By cone duality theory, this is equivalent to stating that the primal cone $K(A)$ possesses a non-empty interior and has dimension m. Since $K^o(A)$ is a pointed polyhedral cone, it is finitely generated and admits a unique collection of extreme rays. Accordingly, it can be expressed as the conic hull of its extreme rays, i.e. $K^o(A) = \text{cone}(r_1, \ldots, r_l)$, where $r_i \in \mathbb{R}^m$ denotes the i-th extreme ray of $K^o(A)$.

We show in what follows that the matrix $E^r \in \mathbb{R}^{l\times m}$ in which each line corresponds to an extreme ray of $K^o(A)$ is such that $K(E^r) = K(A)$.

Theorem 3.1. *Suppose $K^o(A)$ is pointed. Let $\{r_i\}_{i=1}^l$ be the set of its extreme rays, with $l \leq k$.*

The matrix $E^r \in \mathbb{R}^{l\times m}$ defined by $E^r = \begin{pmatrix} r_1^T \\ \vdots \\ r_l^T \end{pmatrix}$ is an effective equivalent of A, i.e., $E^r \sim_e A$.

Proof. First, we show that $K(E^r) \subset K(A)$. To this aim, let us consider a row vector a_i, $i = 1 \ldots k$, of matrix A. Clearly, $a_i \in cone(A)$. By Proposition 3.2, we have that $a_i \in K^o(A)$. Consequently, a_i can be written as a conic combination of the extreme rays of $K^o(A)$. There thus exists a vector $c_i \in \mathbb{R}^l_+$ such that $a_i = c_i^T E^r$.

Hence, the matrix $C \in \mathbb{R}_+^{k \times l}$ defined by $C = \begin{pmatrix} c_1^T \\ \vdots \\ c_l^T \end{pmatrix}$ is such that $CE^r = A$.

We now consider a vector $x \in \mathbb{R}^m$ belonging to $K(E^r)$. We have:

$$E^r x \leq 0 \tag{18a}$$
$$\implies CE^r x \leq 0 \tag{18b}$$
$$\implies Ax \leq 0 \tag{18c}$$

where inequality (18b) holds because matrix C involves only nonnegative coefficients. Inequality (18c) implies that $x \in K(A)$. We conclude that $K(E^r) \subset K(A)$.

Second, we show that $K(A) \subset K(E^r)$. Let r_i, $i = 1 \dots l$ be an extreme ray of $K^o(A)$. By Proposition 3.2, r_i belongs to $cone(A)$, so that it can be written as a conic combination of the row vectors of A. There thus exists a vector $\mu_i \in \mathbb{R}_+^k$ such that $r_i = \mu_i^T A$.

We now consider a vector $x \in \mathbb{R}^m$ belonging to $K(A)$. We have:

$$Ax \leq 0 \tag{19a}$$
$$\implies \mu_i^T Ax \leq 0 \qquad \forall i = 1 \dots l \tag{19b}$$
$$\implies r_i^T x \leq 0 \qquad \forall i = 1 \dots l \tag{19c}$$
$$\implies E^r x \leq 0 \tag{19d}$$

where Inequality (19b) holds because vector μ_i is non-negative and Inequality (19c) is obtained by using the relation $r_i = \mu_i^T A$. Inequality (19d) implies that $x \in K(E^r)$. We conclude that $K(A) \subset K(E^r)$.

Finally, we have $K(E^r) = K(A)$. □

Theorem 3.2 below follows immediately from Theorem 3.1.

Theorem 3.2. *Consider Problem* (1)–(4) *and assume that* $K^o(A)$ *is pointed. Let* $\{r_i\}_{i=1}^l$ *be the finite set of its extreme rays.*

Then, for any $\hat{x} \in \mathcal{X}$*, the indicator Benders cut* $\eta \geq (Z(\hat{x}^i)+M)\mathbb{1}_{Ax \leq A\hat{x}^i} - M$ *is equivalent to:*

$$\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{E^r x \leq E^r \hat{x}} - M \tag{20}$$

where the matrix $E^r \in \mathbb{R}^{l \times m}$ *is defined by* $E^r = \begin{pmatrix} r_1^T \\ \vdots \\ r_l^T \end{pmatrix}$.

Note that, in practice, the set of extreme rays of the finitely generated cone $cone(A) = K^o(A)$ correspond to a subset of rows of A and can be easily identified in a pre-optimization step by running Algorithm 4 provided in Appendix A.

3.4.1 Example

Consider an instance of Problem (1)-(4) in which there are $m = 2$ upper level variables, i.e. $x = (x_1, x_2)$, and the coupling matrix A comprising $k = 4$ rows is given by:

$$A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0.75 & -1 \\ -1 & 0.25 \end{pmatrix} \tag{21}$$

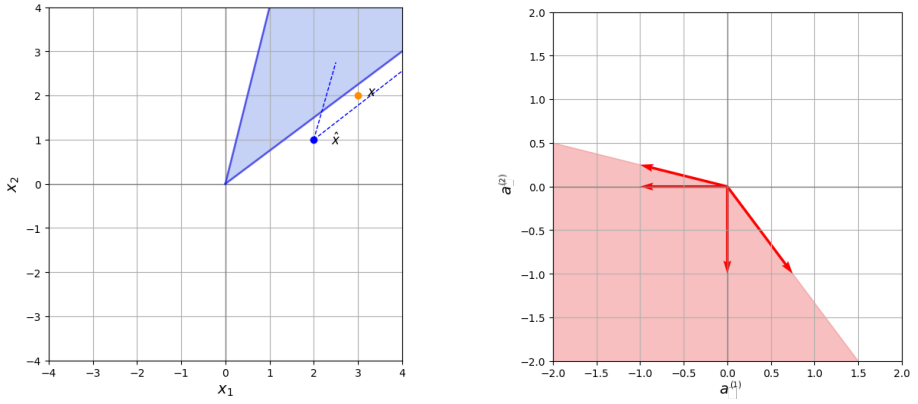
The corresponding cone $K(A)$ is shaded in light blue in Figure 1a. Suppose the point $\hat{x} = (2, 1)$ has already been explored, and the corresponding value $Z(\hat{x})$ is known. As shown in the figure, the point $x = (3, 2)$ lies in the shifted cone $\{\hat{x}\} + K(A)$, i.e., $x - \hat{x} \in K(A)$. Therefore, we can infer that $Z(x) \geq Z(\hat{x})$ without explicitly solving for $Z(x)$.

The cone $K(A)$ is two-dimensional with a non-empty interior, and hence its polar cone $K^o(A)$ is pointed. Figure 1b displays in red the four generator vectors of $K^o(A)$, i.e. the four row vectors of matrix A, in a Euclidean space in which the first and second coordinates correspond respectively to the first component $a^{(1)}$ and the second component $a^{(2)}$) of a row vector of A. The polar cone $K^o(A)$ is shown colored in light red on this figure.

The extreme rays of $K^o(A)$ are $(-1, 0.25)$ and $(0.75, -1)$. Hence, the matrix E^r defined by:

$$E^r = \begin{pmatrix} 0.75 & -1 \\ -1 & 0.25 \end{pmatrix} \tag{22}$$

is an effective equivalent of matrix A involving only $l = 2$ rows.



(a) The cone $K(A) = \{x \mid Ax \leq 0\}$ (b) The polar cone $K^{\circ}(A)$

Figure 1: Cone $K(A)$ with a pointed polar cone $K^o(A)$

3.5 Reduction of coupling constraints: general $K^o(A)$

In general, the polar cone $K^o(A)$ may not be pointed. From cone duality theory, this is equivalent to stating that $K(A)$ has empty interior and is contained within a proper subspace of $\mathbb{R}^m$.

This situation occurs, for example, when both a row vector a^T and its negative $-a^T$ are present in matrix A. In this case, $K(A)$ lies entirely within the subspace of dimension $m-1$ defined by $\{x \in \mathbb{R}^m \mid a^T x = 0\}$. As this subspace has empty interior, so does $K(A)$.

Before constructing an effective equivalent matrix A' in this case, we first identify the implicit equalities in the constraint system $Ax \leq 0$ and show that $K(A)$ has empty interior if and only if the constraint system involves at least one implicit equality.

Let a_i^T denote the i-th row of matrix A. Recall that the inequality $a_i^T x \leq 0$ in the system $Ax \leq 0$ is called an *implicit equality* if it is satisfied as an equality by all solutions of the system, i.e. if $Ax \leq 0$ implies $a_i^T x = 0$ for all $x \in \mathbb{R}^m$.

Let $A^= \in \mathbb{R}^{k^= \times m}$ denote the submatrix of A consisting of all rows that correspond to implicit equalities, and $A^< \in \mathbb{R}^{k^< \times m}$ the remaining rows. For notational completeness, we define $A^= := \mathbf{0}^{1\times m}$ if $A = A^<$, and $A^< := \mathbf{0}^{1\times m}$ if $A = A^=$, where $\mathbf{0}^{1\times m}$ denotes the zero row vector in $\mathbb{R}^m$.

Proposition 3.3. *$K(A)$ has empty interior if and only if $A^= \neq \mathbf{0}^{1\times m}$.*

Proof. We first show that the fact that $K(A)$ has empty interior implies $A^= \neq \mathbf{0}^{1\times m}$. This is done by contradiction.

Let us thus assume that $K(A)$ has empty interior and that $A^= = \mathbf{0}^{1\times m}$. We have $A = A^<$ so that there exist a vector $\tilde{x} \in \mathbb{R}^m$ and a strictly negative vector $\epsilon \in \mathbb{R}^k_-$ such that $A\tilde{x} \leq \epsilon < 0$. Since $A\tilde{x}$ lies in the interior of the negative orthant, there exists an open neighborhood of $A\tilde{x}$, denoted by $B_k(A\tilde{x}) \subseteq \mathbb{R}^k$, containing only negative vectors. As the linear transformation defined by A is continuous, the preimage of $B_k(A\tilde{x})$ by this transformation is contained in an open set containing $\tilde{x}$. There thus exists an open neighborhood of $\tilde{x}$, denoted by $B_m(\tilde{x}) \subseteq \mathbb{R}^k$, such that for all $x \in B_m(\tilde{x})$, $Ax \in B_k(A\tilde{x})$, i.e. $Ax < 0$. This implies that $B_m(\tilde{x}) \subseteq K(A)$, which contradicts with the fact that $K(A)$ has empty interior.

We then prove that $A^= \neq \mathbf{0}^{1\times m}$ implies that $K(A)$ has empty interior.

Assume $A^= \neq \mathbf{0}^{1\times m}$, then by the definition of implicit equalities, we have:

$$Ax \leq 0 \Rightarrow A^= x = 0 \qquad\qquad \forall x \in \mathbb{R}^m \tag{23}$$

which implies that $K(A) \subseteq \{x \in \mathbb{R}^m \mid A^= x = 0\}$. Since this subspace has empty interior in $\mathbb{R}^m$, it follows that $K(A)$ also has empty interior. □

This result implies that in the non-pointed case, the polar cone $K^o(A)$ can be decomposed as the Minkowski sum of two components: a subspace of $\mathbb{R}^m$ and a pointed cone orthogonal to that subspace.

Theorem 3.3. *Let $A^=$ be the submatrix of A consisting of all implicit equalities, and let $A^<$ denote the submatrix consisting of the remaining rows. Then, the polar cone $K^o(A)$ can be decomposed as the Minkowski sum of a subspace and a pointed cone:*

$$K^o(A) = lin(A^=) + proj_{lin(A^=)^\perp} cone(A^<). \tag{24}$$

In particular, if $A = A^=$, then $cone(A^<) = \{0\}$, and if $A = A^<$, then $lin(A^=) = \{0\}$.

To prove Theorem 3.3, we first establish two auxiliary results.

Proposition 3.4. *The cone $cone(A^=)$ is a subspace of $\mathbb{R}^m$. In other words, $cone(A^=) = lin(A^=)$.*

Proof. Note that $cone(A^=) \subseteq lin(A^=)$ holds as the set of conic combinations of rows of $A^=$ is obviously included in the set of linear combinations of these rows.

We now prove that $lin(A^=) \subseteq cone(A^=)$. To this aim, we first show that $cone(A^=)$ is a subspace of $\mathbb{R}^m$.

Let us first consider the cone $K(A^=)$ defined by $K(A^=) := \{x \in \mathbb{R}^m \mid A^=x \leq 0\}$. By the definition of implicit equalities, we have $K(A^=) = \{x \in \mathbb{R}^m \mid A^=x = 0\}$. $K(A^=)$ is the solution set of an homogeneous system of linear equations and is thus a subspace of $\mathbb{R}^m$.

Furthermore, we have that, for any $x \in K(A^=)$, $-x \in K(A^=)$ also holds. Consequently, for any $x \in K(A^=)$ and any $\lambda \in K^o(A^=)$, both $\lambda^T x \leq 0$ and $-\lambda^T x \leq 0$ must hold, which implies $\lambda^T x = 0$. This means that $K^o(A^=)$ is the orthogonal complement of $K(A^=)$ and is therefore also a subspace of $\mathbb{R}^m$.

By Proposition 3.2, $cone(A^=) = K^o(A^=)$, which shows that $cone(A^=)$ is a subspace of $\mathbb{R}^m$ containing the rows of $A^=$. Since $lin(A^=)$ is the smallest subspace of $\mathbb{R}^m$ containing the rows of $A^=$, we have that $lin(A^=) \subseteq cone(A^=)$. This concludes the proof. □

Proposition 3.5. *The cone $proj_{lin(A^=)^\perp} cone(A^<)$ is pointed.*

Proof. The proof is done by contradiction. Suppose that $\text{proj}_{lin(A^=)^\perp} cone(A^<)$ is not pointed. Then, there exist two nonzero vectors λ_1, λ_2 in this projected cone such that $\lambda_1 + \lambda_2 = 0$.

Let $v_1, v_2 \in cone(A^<)$ be preimages of λ_1, λ_2 respectively. By linearity of the orthogonal projection operator, we have:

$$\begin{aligned} & proj_{lin(A^=)^\perp}(v_1 + v_2) \\ =& proj_{lin(A^=)^\perp}(v_1) + proj_{lin(A^=)^\perp}(v_2) \\ =& \lambda_1 + \lambda_2 \\ =& 0. \end{aligned}$$

This implies that $v_1 + v_2$ is orthogonal to $lin(A^=)^\perp$ and thus belongs to $lin(A^=)$.

Furthermore, $v_1 + v_2$ also belongs to $cone(A^<)$ and can thus be written as a conic combination of the rows of $A^<$. Consequently, there exists $u \in \mathbb{R}^k_+$ such that $v_1 + v_2 = u^T A^<$, i.e. there exists $u \in \mathbb{R}^k_+$ such that $u^T A^< \in lin(A^=)$.

Note that, for any vector $w \in lin(A^=)$, we have $w^T x = 0$ for all $x \in K(A)$. Since $u^T A^< \in lin(A^=)$, we have $u^T A^< x = 0$ for all $x \in K(A)$. However, by construction, there exists $x' \in K(A)$ such that $A^< x' < 0$, and hence $u^T A^< x' < 0$, leading to a contradiction.

Thus, the projection $\text{proj}_{lin(A^=)^\perp}\ cone(A^<)$ is pointed. □

Proof of Theorem 3.3. Since $lin(A^=)$ is a subspace of $\mathbb{R}^m$, the polar cone $K^o(A)$ can be decomposed as:

$$\begin{aligned} K^o(A) \quad &= cone(A) \\ &= proj_{lin(A^=)} cone(A) + proj_{lin(A^=)^\perp}\ cone(A) \end{aligned}$$

By Proposition 3.4, we have $lin(A^=) = cone(A^=) \subseteq cone(A)$, namely, $cone(A)$ contains the whole subspace $lin(A^=)$. Therefore, $proj_{lin(A^=)} cone(A) = lin(A^=)$

For the second term, since $cone(A)$ is the conic combinations of row vectors, it can also be expressed as the Minkowski sum of $cone(A^=)$ and $cone(A^<)$, i.e., $cone(A) = cone(A^=) + cone(A^<) = lin(A^=) + cone(A^<)$. It follows that $proj_{lin(A^=)^\perp}\ cone(A) = proj_{lin(A^=)^\perp}\ cone(A^<)$. By Proposition 3.5, the cone $proj_{lin(A^=)^\perp}\ cone(A^<)$ is pointed. □

Note that if $A = A^<$, i.e., no implicit equality appears in the coupling constraints, then $lin(A^=) = \{0\}$, and we recover the pointed case discussed in Section 3.4.

Proposition 3.6. *Let $\{b_i\}_{i=1}^{l^=}$ be a basis of the subspace $lin(A^=)$, and $\{r_i\}_{i=1}^{l^<}$ be the set of extreme rays of the pointed cone $proj_{lin(A^=)^\perp}\ cone(A^<)$. We define two matrices $E^b \in \mathbb{R}^{l^= \times m}$ and $E^r \in \mathbb{R}^{l^< \times m}$ as follows:*

$$E^b = \begin{pmatrix} b_1^T \\ \vdots \\ b_{l^=}^T \end{pmatrix}, E^r = \begin{pmatrix} r_1^T \\ \vdots \\ r_{l^<}^T \end{pmatrix} \tag{25}$$

We have:

$$A \sim_e \begin{pmatrix} E^b \\ -E^b \\ E^r \end{pmatrix}. \tag{26}$$

Proof.

$$K^o\left(\begin{pmatrix} E^b \\ -E^b \\ E^r \end{pmatrix}\right) = cone\left(\begin{pmatrix} E^b \\ -E^b \end{pmatrix}\right) + cone(E^r) \tag{27}$$

$$= lin(E^b) + cone(E^r) \tag{28}$$

$$= lin(A^=) + proj_{lin(A^=)^\perp}\ cone(A^<) \tag{29}$$

$$= K^o(A) \tag{30}$$

□

In the effective equivalent form of A which is expressed in block matrix in (26), the matrices E^b and $-E^b$ both appear in the row blocks, implying that they correspond to a system of equations. More precisely:

$$K(A) = \{x|Ax \leq 0\} = \{x|E^b x = 0, E^r x \leq 0\} \tag{31}$$

The corresponding indicator Benders cut involves a disjunction of both inequalities and equalities.

Theorem 3.4. *The indicator Benders cut of the form*

$$\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{Ax \leq A\hat{x}} - M \tag{32}$$

can be equivalently expressed as:

$$\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{(E^b x = E^b \hat{x}) \wedge (E^r x \leq E^r \hat{x})} - M \tag{33}$$

where the rows of E^b form a basis of $lin(A^=)$, and the rows of E^r are the extreme rays of $proj_{lin(A^=)^\perp}\ cone(A^<)$.

The proof follows directly from Proposition 3.6.

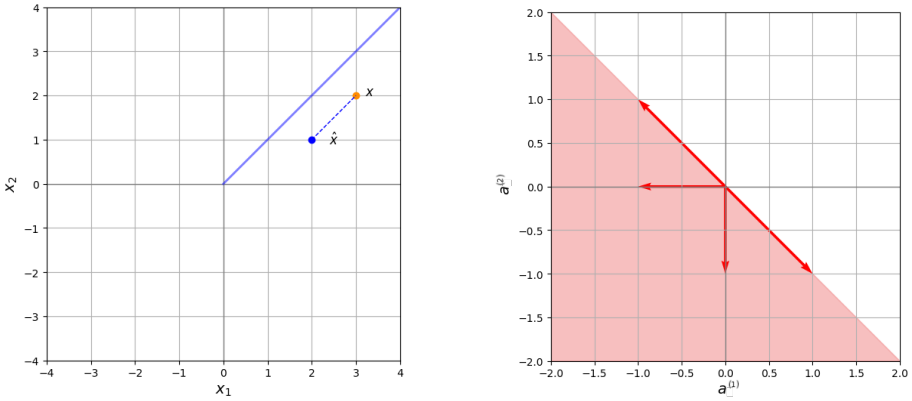
The indicator function now depends on the conjunction of an equality system and an inequality system. The block E^b and E^r are computed prior to solving the Benders decomposition iteratively. The basis E^b can be obtained from A using Algorithm 2 in Appendix A. Once E^b is determined, we apply Algorithm 3 in Appendix A to obtain $proj_{lin(E^b)^\perp} cone(A)$, which are the rows of $A^<$ projected onto the orthogonal complement of $lin(A^=)$. The matrix E^r of extreme rays can then be computed using Algorithm 4 in Appendix A.

3.5.1 Example

Consider an instance of Problem (1)-(4) in which there are $m = 2$ upper level variables, i.e. $x = (x_1, x_2)$, and the coupling matrix A comprising $k = 4$ rows is given by:

$$A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{34}$$

The cone $K(A)$ is illustrated in Figure 2a. In this case, $K(A)$ is a ray with an empty interior. Suppose the point $\hat{x} = (2, 1)$ has already been explored so that the value of $Z(\hat{x})$ is known. The point $x = (3, 2)$ lies on the ray $\{\hat{x}\}+K(A)$, so that $x - \hat{x} \in K(A)$. Therefore, we can conclude that $Z(x) \geq Z(\hat{x})$ without explicitly solving for $Z(x)$.



(a) The cone $K(A) = \{x \mid Ax \leq 0\}$ (b) The polar cone $K^{\circ}(A)$

Figure 2: Cone $K(A)$ with a non-pointed polar cone $K^o(A)$

The polar cone $K^o(A)$ is shown in Figure 2b. This cone is a half-space. It contains a subspace whose direction is given by the vector $(-1, 1)$ and is thus not pointed. Consequently, it cannot be generated by a conic combination of a finite set of extreme rays.

We thus partition matrix A as follows:

$$A^{=} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \qquad A^{<} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \tag{35}$$

The cone $cone(A^{<})$ is a pointed cone with two extreme rays. We can further reduce its representation by projecting it onto the orthogonal complement of the subspace $lin(A^{=})$. This yields the blocks:

$$E^b = \begin{pmatrix} 1 & -1 \end{pmatrix} \qquad E^r = \begin{pmatrix} -1 & -1 \end{pmatrix} \tag{36}$$

From this, we obtain the following equivalence:

$$Ax \leq A\hat{x} \Leftrightarrow \begin{cases} E^b x = E^b \hat{x} \\ E^r x \leq E^r \hat{x} \end{cases} \tag{37}$$

3.6 Effectiveness of indicator Benders cuts

In the previous sections, we introduced the concept of effective equivalent between matrices. However, even an effective equivalent matrix may still have a

large number of rows, making it computationally expensive to formulate and use in practice. In this section, we present an alternative approach that allows for the use of *less effective* but easier to formulate indicator Benders cuts.

Let $\mathcal{M}^m$ denote the set of all real-valued matrices with m columns. We now extend the notion of effectiveness to define a partial order on $\mathcal{M}^m$.

Definition 3.2. Let $A^1, A^2 \in \mathcal{M}^m$. We say that A^1 is *less effective* than A^2 if:

$$A^1 x \leq 0 \implies A^2 x \leq 0 \tag{38}$$

and denote this as $A^1 \preceq_e A^2$.

Note that A^1 and A^2 do not necessarily have the same number of rows.

Proposition 3.7. *For two matrices $A^1, A^2 \in \mathcal{M}^m$, the following statements are equivalent:*

1. $A^1 \preceq_e A^2$
2. $K(A^1) \subseteq K(A^2)$
3. $K^o(A^1) \supseteq K^o(A^2)$
4. $A^1 x \leq 0 \implies A^2 x \leq 0$
5. $\mathbb{1}_{A_1 x \leq A_1 \hat{x}} \leq \mathbb{1}_{A_2 x \leq A_2 \hat{x}}$

Proposition 3.8. *The relation $\preceq_e$ defines a partial order on $\mathcal{M}^m$.*

The relation $\preceq_e$ has the property of reflexivity, antisymmetry and transitivity. It extends the effective equivalent: if $A^1 \preceq_e A^2$ and $A^2 \preceq_e A^1$, then $A^1 \sim_e A^2$.

Theorem 3.5. *Let $\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{Ax \leq A\hat{x}} - M$ be an indicator Benders cut. If there exists a matrix $A' \in \mathcal{M}^m$ such that $A' \preceq_e A$, then the alternative cut:*

$$\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{A'x \leq A'\hat{x}} - M \tag{39}$$

is also a strong Benders cut and guarantees the convergence of the Benders decomposition algorithm. We refer to this as a less effective cut.

In situations where the effective equivalent of A contains too many rows, we may choose a matrix A' with fewer rows such that $A' \preceq_e A$ to define the indicator cut. This results in a cut that is 'strong' in the sense of guaranteeing convergence. However, such a cut has a smaller effective region: it only improves the lower bound for points x such that $x - \hat{x} \in K(A') \subseteq K(A)$.

Proposition 3.9 (Non-negative matrix A)**.** *Let $A \in \mathbb{R}_+^{k \times m}$ have only non-negative entries. Then the $m \times m$ identity matrix I^m satisfies:*

$$I^m \preceq_e A.$$

Proof. Let $x^1, x^2 \in \mathbb{R}^m$ with $x^1 \leq x^2$ elementwise. For any $a \in \mathbb{R}^m_+$, we have $a^T x^1 \leq a^T x^2$, which implies $Ax^1 \leq Ax^2$. □

The monotone cut proposed by [Liu et al., 2024] is a direct consequence of Proposition 3.9. When A is non-negative, $I^m \preceq_e A$, so that $x_1 \geq x_2$ implies $Ax_1 \geq Ax_2$ and thus $Z(x_1) \leq Z(x_2)$ for all $x_1, x_2 \in \mathcal{X}$. Therefore, the cut induced by I^m is valid.

Although the effectiveness $\preceq_e$ is a partial order relation of the set $\mathcal{M}^m$, it has a minimum element in the set $\mathcal{M}^m$.

Proposition 3.10 (Minimum effective matrix)**.** *Let $I^m \in \mathcal{M}^m$ denote the identity matrix of m columns, then the matrix $\begin{pmatrix} I^m \\ -I^m \end{pmatrix}$ is the minimum element in $\mathcal{M}^m$ with respect to the partial order relation $\preceq_e$. That is, for any matrix $A \in \mathcal{M}^m$,*

$$\begin{pmatrix} I^m \\ -I^m \end{pmatrix} \preceq_e A \tag{40}$$

Proof. We first compute the polar cone:

$$K^o(\begin{pmatrix} I^m \\ -I^m \end{pmatrix}) = cone(I^m) + cone(-I^m) \tag{41}$$

$$= lin(I^m) \tag{42}$$

$$= \mathbb{R}^m \tag{43}$$

By Proposition 3.7, since $K^o(\begin{pmatrix} I^m \\ -I^m \end{pmatrix})$ is the entire space of $\mathbb{R}^m$, the polar cone $K^o(A)$ of any matrix $A \in \mathcal{M}^m$ is contained in $K^o(\begin{pmatrix} I^m \\ -I^m \end{pmatrix})$, implying that $\begin{pmatrix} I^m \\ -I^m \end{pmatrix}$ is the least effective with respect to $\preceq_e$. □

This result implies that for any problem of the form (1)–(4), an indicator cut defined by $\begin{pmatrix} I^m \\ -I^m \end{pmatrix}$ is always applicable, though it only improves the lower bound for the current solution $\hat{x}$ itself.

Corollary 3.1. *The no-good cut (also known as the L-shaped or combinatorial cut) is the least effective indicator Benders cut for all problems of the form* (1)–(4)*.*

Proof. A no-good cut can be written as:

$$\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{x=\hat{x}} - M \tag{44}$$

By the cone decomposition introduced above, we have $A^= = I^m$ and $A^< = \mathbf{0}^{1\times m}$, where I^m is the identical matrix and $\mathbf{0}^{1\times m}$ is the zero matrix. By Proposition 3.10, we conclude that the no-good cut is the least effective indicator Benders cut. □

This result explains why the no-good cut is universally applicable: its associated cone $K(A)$ is inclusion-wise minimal among the cones generated by all matrices with m columns. However, this also implies that its capacity to improve the lower bound is very limited, as it is the least effective indicator cut and only provides a lower bound at the solution used to generate it.

The effectiveness order introduced above can also be used to interpret related matrix aggregation procedures beyond indicator Benders cuts. In particular, [Yıldız et al., 2022] propose to multiply the linking system by a non-negative matrix in order to reduce the row number of an inequality system. More precisely, given a non-negative matrix C, the original system $Ax \leq A\hat{x}$ is replaced by the relaxed system $CAx \leq CA\hat{x}$.

Since C has non-negative entries, any vector satisfying $Ax \leq A\hat{x}$ also satisfies $CAx \leq CA\hat{x}$. Therefore, in the sense of Definition 3.2, the aggregated matrix CA is generally more effective than A. Our notion of effectiveness thus provides a unified way to compare different choices of aggregation matrices C. Moreover, if C is chosen so that $CA \sim_e A$, then the aggregated system is no longer a surrogate relaxation but an exact reformulation of the original one.

4 Mathematical formulation of indicator Benders cut

Due to the non-convex nature of the indicator function, incorporating indicator Benders cuts into the master problem formulation is significantly more challenging than generating classical linear Benders cuts derived from the dual of LP subproblems. In this section, we investigate how to formulate the indicator Benders cuts introduced in Section 3 when the master problem is a pure integer linear program.

An important modeling consideration is the handling of subproblem infeasibility. Since MILP formulations cannot directly represent $+\infty$ values, we incorporate feasibility cuts to exclude infeasible upper-level solutions, in addition to the standard optimality cuts.

Given an effective equivalent block matrix $\begin{pmatrix} E^b \\ -E^b \\ E^r \end{pmatrix}$ of the coupling matrix A, the indicator Benders cuts considered in this section are given by:

$$\eta \geq (Z(\hat{x}) + M)\mathbb{1}_{(E^b x = E^b \hat{x}) \wedge (E^r x \leq E^r \hat{x})} - M \tag{45}$$

$$\mathbb{1}_{(E^b x = E^b \hat{x}) \wedge (E^r x \leq E^r \hat{x})} = 0 \tag{46}$$

Constraint (45) is an optimality cut in which M is a sufficiently large positive constant. This constraint imposes a lower bound on the value function $Z(\cdot)$ over a subset of the master problem feasible region. Equality (46) is a feasibility cut forbidding infeasible master problem solutions.

The rest of this section discusses two ways to express indicator cuts using binary variables: a general MILP formulation based on [Liu et al., 2024] that

adds binary variables at each iteration (Section 4.1), and an alternative approach derived from [Forbes et al., 2024] that a priori adds all required binary variables before the BD algorithm starts (Section 4.2).

4.1 Formulation of indicator Benders cut: binary expansion

The optimality cut (45) can be interpreted as a logical implication:

$$\text{if } (E^b x = E^b \hat{x}) \wedge (E^r x \leq E^r \hat{x}), \text{ then } \eta \geq Z(\hat{x}) \tag{47}$$

This implication can be reformulated in disjunctive logic form as:

$$\neg[(E^b x = E^b \hat{x}) \wedge (E^r x \leq E^r \hat{x})] \vee (\eta \geq Z(\hat{x})) \tag{48}$$

$$\implies \neg(E^b x = E^b \hat{x}) \vee \neg(E^r x \leq E^r \hat{x}) \vee (\eta \geq Z(\hat{x})) \tag{49}$$

$$\implies \bigvee_{i=1}^{l^=} (e_i^b x \neq e_i^b \hat{x}) \vee \bigvee_{i=1}^{l^<} (e_i^r x > e_i^r \hat{x}) \vee (\eta \geq Z(\hat{x})) \tag{50}$$

The indicator optimality cut thus corresponds to a disjunction of three terms: a disjunctive of non-equality constraints, a disjunctive of strict inequalities and a single linear inequality involving η.

Since the matrices E^r and E^b are composed of extreme rays of a pointed cone and basis vectors of a subspace, respectively, their entries can be chosen to be integers. Furthermore, recall that x is an integer decision vector. Consequently, the strict inequality $e_i^r x > e_i^r \hat{x}$ can be replaced by $e_i^r x \geq e_i^r \hat{x} + 1$.

If $E^b \neq \mathbf{0}^{1\times m}$, i.e., the non-equality term of the disjunction must be modeled in the master problem, we perform a binary expansion of $e_i^b x$. Let $\underline{z}_i^b$ and $\bar{z}_i^b$ denote the lower and upper integer bounds of $e_i^b x$, i.e., $\underline{z}_i^b \leq e_i^b x \leq \bar{z}_i^b$. The binary expansion is given by:

$$e_i^b x = \sum_{j=0}^{j_i^{max}} \xi_{i,j}^b 2^j + \underline{z}_i^b \tag{51}$$

where $\xi_{i,j}^b$ is a binary variable indicating whether the term 2^j appears in the binary expansion of $e_i^b x - \underline{z}_i^b$. The number of such binary expansion variables depends on the bounds of $e_i^b x$, we thus have $j_i^{max} = \lceil \log_2(\bar{z}_i^b - \underline{z}_i^b) \rceil$.

Let $\{\hat{\xi}_{i,j}^b\}_{j=0}^{j_i^{max}}$ denote the binary expansion of $e_i^b \hat{x}$, for all $i \in \{1, \cdots, l^=\}$. We define the two following index sets:

$$J_0^= = \{(i,j) | \hat{\xi}_{i,j}^b = 0, i \in \{1, \cdots, l^=\}, j \in \{0, \cdots, j_i^{max}\}\} \tag{52}$$

$$J_1^= = \{(i,j) | \hat{\xi}_{i,j}^b = 1, i \in \{1, \cdots, l^=\}, j \in \{0, \cdots, j_i^{max}\}\} \tag{53}$$

We also introduce a binary variable $\alpha^b \in \{0,1\}$ such that $\alpha^b = 1$ indicates that at least one equality $e_i^b x = e_i^b \hat{x}$, for $i \in \{1, \cdots, l^=\}$, is violated, while $\alpha^b = 0$ imposes no restriction on these equalities. We then add Constraint (54) to the master problem formulation. This constraint enforces that, if $\alpha^b = 1$, then at

least one binary expansion variable $\xi^b_{i,j}$ differs from its counterpart $\hat{\xi}^b_{i,j}$ in the binary expansion associated with the current solution.

$$\sum_{(i,j)\in J_0^=} \xi^b_{i,j} + \sum_{(i,j)\in J_1^=} (1-\xi^b_{i,j}) \geq \alpha^b \tag{54}$$

Next, we address the strict inequality $\bigvee_{i=1}^{l^<}(e_i^r x > e_i^r \hat{x})$, which can be equivalently written in integer settings as $\bigvee_{i=1}^{l^<}(e_i^r x \geq e_i^r \hat{x}+1)$. For each $i \in \{1,...,l^<\}$, we introduce a binary variable α_i^r to indicate whether this inequality is activated. Specifically, when $\alpha_i^r = 1$, the inequality $e_i^r x \geq e_i^r \hat{x}+1$ is enforced; when $\alpha_i^r = 0$, the inequality may or may not hold. The condition is formulated as:

$$e_i^r x \geq (e_i^r \hat{x}+1)\alpha_i^r - M_i(1-\alpha_i^r) \quad \forall i \in \{1,\ldots,l^<\} \tag{55}$$

where M_i is a sufficiently large constant. Let $\underline{z}_i^r$ denote the lower bound of $e_i^r x$. If $\underline{z}_i^r$ is known, we can take $M_i = -\underline{z}_i^r$.

The final term in the disjunction, $\eta \geq Z(\hat{x})$, is modeled using a binary variable $\alpha_0 \in \{0,1\}$:

$$\eta \geq Z(\hat{x})\alpha_0 - M_0(1-\alpha_0) \tag{56}$$

where $-M_0$ is a valid lower bound on η, ensuring the inequality holds regardless of the value of α_0.

Overall, incorporating indicator cuts into the master problem requires introducing, in advance, a collection of binary variables ξ^b and the associated constraints (51) to represent the binary expansion of each quantity $e_i^b x$. Furthermore, a set of binary variables α and their corresponding constraints must be added whenever an indicator cut is generated during the iterative generalized Benders decomposition algorithm.

More precisely, in case an indicator optimality cut has to be generated, we rely on the disjunctive approach proposed by [Balas, 2018] and add the following constraints to the master problem formulation:

$$\eta \geq Z(\hat{x})\alpha_0 - M_0(1-\alpha_0) \tag{57}$$

$$\sum_{(i,j)\in J_0^=} \xi^b_{i,j} + \sum_{(i,j)\in J_1^=} (1-\xi^b_{i,j}) \geq \alpha^b \tag{58}$$

$$e_i^r x \geq (e_i^r \hat{x}+1)\alpha_i^r - M_i(1-\alpha_i^r) \qquad \forall i \in \{1,\cdots,l^<\} \tag{59}$$

$$\alpha_0 + \alpha^b + \sum_{i=1}^{l^<} \alpha_i^r = 1 \tag{60}$$

$$\alpha_0, \alpha^b, \alpha_i^r \in \{0,1\} \tag{61}$$

In case an indicator feasibility cut has to be generated, a similar set of constraints is used. The only difference comes for the fact the inequality on η is not needed anymore so that variable α_0 is not required. The resulting MILP formulation is given by:

$$\sum_{(i,j)\in J_0^=} \xi_{i,j}^b + \sum_{(i,j)\in J_1^=} (1-\xi_{i,j}^b) \geq \alpha^b \tag{62}$$

$$e_i^r x \geq (e_i^r \hat{x} + 1)\alpha_i^r - M_i(1-\alpha_i^r) \qquad \forall i \in \{1, \cdots, l^<\} \tag{63}$$

$$\alpha^b + \sum_{i=1}^{l^<} \alpha_i^r = 1 \tag{64}$$

$$\alpha^b, \alpha_i^r \in \{0,1\} \tag{65}$$

Generating an indicator cut thus requires adding new binary variables to the master problem formulation over the course of Benders decomposition algorithm. This feature makes it difficult to apply indicator cuts within a Branch-and-Benders-Cut framework, as the introduction of new auxiliary binary variables during the branching process is difficult to handle.

4.2 Formulation of indicator Benders cut: unary expansion

The method presented in Section 4.1 requires introducing $l^<+2$ auxiliary binary variables at each iteration. However, when $E^r x$ take values in a relatively small integer range, there is an alternative approach, inspired by the logic-based Benders cut technique introduced by [Forbes et al., 2024]. This approach allows to formulate the indicator Benders cuts without adding new binary variables in each iteration.

Without loss of generality, we assume that $\underline{z}_i^r \leq e_i^r x \leq \bar{z}_i^r$ for all $i \in \{1, \ldots, l^<\}$, where $\underline{z}_i^r$ and $\bar{z}_i^r$ denotes a known lower bound and upper bound on $e_i^r x$, respectively. Each $e_i^r x$ can then be represented by a sequence of binary variables (also called one-hot encoding or unary expansion) as follows:

$$e_i^r x = \sum_{j=\underline{z}_i^r}^{\bar{z}_i^r} \zeta_{i,j}^r \cdot j \tag{66}$$

where $\zeta_{i,j}^r = 1$ if $e_i^r x = j$, and $\zeta_{i,j}^r = 0$ otherwise. The variables $\zeta_{i,j}^r$ satisfy the following constraints:

$$\sum_{j=\underline{z}_i^r}^{\bar{z}_i^r} \zeta_{i,j}^r = 1 \qquad \forall i \in \{1, \cdots, l^<\} \tag{67}$$

Similarly, we introduce a sequence of binary variables $\zeta_{i,j}^b \in \{0,1\}$ to carry out a unary expansion of each vector $e_i^b x$ taking values between the lower and upper bound $\underline{z}_i^b$ and $\bar{z}_i^b$.

For a given value of $E^r \hat{x}$ and $E^b \hat{x}$, we define the following index sets:

$$L_0^< = \{(i,j) | j < e_i^r \hat{x}\} \qquad L_0^= = \{(i,j) | j < e_i^b \hat{x}\} \tag{68}$$

$$L_1^< = \{(i,j) | j > e_i^r \hat{x}\} \qquad L_1^= = \{(i,j) | j > e_i^b \hat{x}\} \tag{69}$$

The indicator Benders optimality cut can then be written as:

$$\eta \geq (Z(\hat{x}) + M)\Bigg[1 - \sum_{(i,j)\in L_1^<} \zeta_{i,j}^r - \sum_{(i,j)\in L_0^=} \zeta_{i,j}^b - \sum_{(i,j)\in L_1^=} \zeta_{i,j}^b\Bigg] - M \tag{70}$$

and the corresponding feasibility cut takes the form:

$$\sum_{(i,j)\in L_1^<} \zeta_{i,j}^r + \sum_{(i,j)\in L_0^=} \zeta_{i,j}^b + \sum_{(i,j)\in L_1^=} \zeta_{i,j}^b \geq 1 \tag{71}$$

This formulation is logically equivalent to the general approach introduced in Section 4.1, but avoids introducing new auxiliary binary variables at each iteration and reduces the number of additional constraints. However, it requires one auxiliary binary variable for each value in the integer range of each upper-level variable. If the number of such variables is large or their feasible ranges are wide, this transformation may result in a large number of binary variables.

We note that, in the special case where $J_0^= = J_1^= = \emptyset$, this formulation reduces to the one proposed in [Forbes et al., 2024].

5 Generalized Benders decomposition algorithm based on indicator cuts

This section discusses two algorithmic enhancements aimed at improving the numerical time-efficiency of the algorithm and presents our overall generalized BD algorithm based on indicator cuts.

5.1 Generating classical Benders cuts from the linear relaxation of the subproblem

Although the indicator Benders cuts introduced in Section 3 are strong and guarantee theoretical finite convergence to the optimum, relying exclusively on them can lead to poor computational performance. The main reason is that, for a given master problem solution $\hat{x} \in \mathcal{X}$, an indicator cut tightens the lower bound within the cone $\{\hat{x}\} + K(A)$ based on the value $Z(\hat{x})$. If the initial iterate $\hat{x}$ is of low quality, so is $Z(\hat{x})$, resulting in a negligible bound improvement. Consequently, the algorithm may behave no better than a pure enumeration.

Moreover, generating an indicator cut requires solving one or several MILP subproblems. This effort may be wasteful when the current iterate $\hat{x}$ is already infeasible or suboptimal with respect to the linear relaxation of the subproblem. In the first case, it suffices to add a classical linear Benders feasibility cut derived from the LP relaxation, which excludes the infeasible point $\hat{x}$. In the second case, a linear optimality cut obtained from the LP dual of the relaxed subproblem can

raise the lower bound, regardless of whether $\hat{x}$ is feasible for the original MILP subproblem. Therefore, augmenting the algorithm with linear cuts extracted from the linear relaxation of the subproblem provides a simple yet efficient way to accelerate convergence and to reduce overall computational effort.

5.2 Multi-cut implementation of the generalized Benders algorithm

In certain problem settings, once a candidate solution $\hat{x}$ is fixed, the subproblem $\mathbf{SP}(\hat{x})$ can be decomposed into a family of smaller independent optimization problems, denoted by $\mathbf{SP}_i(\hat{x})$ for $i \in I$. These subproblems may differ in the formulation of their coupling constraints. Let A_i denote the coupling matrix associated with subproblem $i \in I$. We can then construct, for each subproblem, a dedicated strong dual function f_i based on A_i for the value function $Z_i(\cdot)$, and consequently generate one indicator cut per subproblem. Thus, multiple indicator cuts may be added to the master problem at each iteration of the generalized Benders decomposition algorithm.

These cuts are usually tighter than a single aggregate cut summarizing the information from all subproblems in one constraint. As a result, the algorithm often reaches optimality in fewer iterations. However, this approach significantly increases the size of the master problem, due to the additional binary variables and constraints required to formulate each cut, which may slow down the solution of the master problem itself. By contrast, a single-cut implementation keeps the master problem more compact, but typically yields a slower improvement of the lower bound at each iteration, thereby increasing the total number of iterations required for convergence.

The choice between a single-cut and a multi-cut implementation therefore requires a careful analysis of the trade-off between cut tightness and computational tractability of the master problem.

5.3 Overall algorithm

A multi-cut implementation of the proposed generalized Benders decomposition algorithm is presented in Algorithm 1.

Lines 2-6 correspond to a pre-processing step in which, for each sub-problem $i \in I$, an effective equivalent of the coupling matrix A_i is constructed and matrices E_i^b and E_i^r are identified. This analysis relies on Algorithms 2, 3, and 4 provided in Appendix A. Line 3 computes a basis of the subspace of $K^o(A_i)$, thereby providing E_i^b. Line 4 projects the rows of A_i onto the orthogonal complement of $lin(E^b) = lin(A^=)$ and constructs the matrix $\tilde{A}_i = proj_{lin(E^b)^\perp}(cone(A)) = proj_{lin(E^b)^\perp}(cone(A^<))$ from these projected rows. Line 5 determines the extreme rays of the pointed cone $K^o(\tilde{A}_i) = cone(\tilde{A}_i)$.

Lines 8-40 describe the iterative process. At each iteration k, the algorithm first solves the current master problem MP^k and updates the lower bound accordingly: see lines 10-11.

Lines 12-21 correspond to the generation of classical Benders cuts from the linear relaxation of the subproblems. For each $i \in I$, the algorithm solves the linear relaxation $\widetilde{SP}_i(\hat{x}^k)$. If $\widetilde{SP}_i(\hat{x}^k)$ is infeasible, a classical Benders feasibility cut is added to the master problem and the current iteration is terminated. If its optimal value is below the current estimate η_i^k in the master problem, a classical Benders optimality cut, which may not be strong, is added.

Lines 22-33 correspond to the generation of indicator Benders cuts. For each $i \in I$, the algorithm solves $SP_i(\hat{x}^k)$ as an MILP. If $SP_i(\hat{x}^k)$ is infeasible, an indicator feasibility cut is generated and added to the master problem using one of the linear reformulations introduced in Section 4, and the current iteration is terminated. If $SP_i(\hat{x}^k)$ is feasible, an indicator optimality cut is generated using one of the formulations introduced in Section 4.

Lines 34-39 correspond to the update of the upper bound and the incumbent solution whenever the current iteration yields a feasible solution better than the incumbent.

The algorithm terminates when the gap between the lower and upper bounds falls below a prescribed tolerance ϵ.

6 Numerical results

To assess the performance of our generalized Benders decomposition algorithm, we carried out numerical experiments on a real-life energy system design problem. This section presents the corresponding results.

6.1 Optimal multi-phase design problem of a cooling system

A cooling system can be described as a decentralized energy infrastructure aimed at satisfying the cooling needs, such as air-conditioning, of nearby buildings. We address the long-term design optimization of a centralized cooling plant over a multi-year planning horizon. The problem mainly consists of selecting, among several technological options, the cooling production and storage equipment to be installed within the plant. These options include standard chillers, ice chillers, and ice storage units. Standard chillers directly produce chilled water to meet instantaneous cooling demand, whereas ice chillers operate by either producing chilled water or generating ice which can be stored and later melted to produce cooling. For each chiller category, the decision process involves choosing the specific models from a list proposed by manufacturers and determining the number of units of each model to be installed. For the storage units, it involves fixing the total ice storage capacity. The system relies on electricity purchased from an external supplier to operate the chillers. The system must establish a contractual agreement with the electricity provider that specifies the maximum instantaneous power that can be drawn from the grid. Consequently, determining this contracted power level is a part of the design optimization problem.

Algorithm 1: Indicator Benders decomposition algorithm

Data: Master problem MP^0, a family of subproblems $\{SP_i(x)\}_{i\in I}$, tolerance $\epsilon > 0$, coefficient matrices of coupling constraints $\{A_i\}_{i\in I}$.
Result: Best feasible solution (x^*, y^*), upper bound UB, and lower bound LB.

for $i \in I$ **do**
  Take matrix A_i as input and use Algorithm 2 to obtain matrix E_i^b;
  Take matrices A_i and E_i^b as input and use Algorithm 3 to obtain matrix $\tilde{A}_i$;
  Take matrix $\tilde{A}_i$ as input and use Algorithm 4 to obtain matrix E_i^r;
end
$k \leftarrow 0$;
$LB \leftarrow -\infty$, $UB \leftarrow +\infty$;
$(x^*, y^*) \leftarrow$ `null`;
while $|UB - LB| > \epsilon$ **do**
  `isFeasible` $\leftarrow$ `True`;
  Solve MP^k and obtain $\hat{x}^k$ and $\hat{\eta}_i^k$, for all $i \in I$;
  $LB \leftarrow c^\top \hat{x}^k + \sum_{i\in I} \hat{\eta}_i^k$;
  for $i \in I$ **do**
    Solve the linear relaxation $\widetilde{SP}_i(\hat{x}^k)$;
    if $\widetilde{SP}_i(\hat{x}^k)$ *is not feasible* **then**
      `isFeasible` $\leftarrow$ `False`;
      Add a linear Benders feasibility cut to MP^k;
      break;
    else
      Add a linear Benders optimality cut to MP^k;
    end
  end
  if *isFeasible* **then**
    for $i \in I$ **do**
      Solve $SP_i(\hat{x}^k)$ and obtain $Z_i(\hat{x}^k)$ and $\hat{y}_i^k$;
      if $SP_i(\hat{x}^k)$ *is not feasible* **then**
        `isFeasible` $\leftarrow$ `False`;
        Add $\mathbb{1}_{(E_i^r x \le E_i^r \hat{x}^k)\wedge(E_i^b x = E_i^b \hat{x}^k)} = 0$ to MP^k;
        break;
      else
        Add $\eta_i \ge (Z_i(\hat{x}^k) + M)\mathbb{1}_{(E_i^r x \le E_i^r \hat{x}^k)\wedge(E_i^b x = E_i^b \hat{x}^k)} - M$ to MP^k;
      end
    end
  end
  if *isFeasible and* $UB > c^\top \hat{x}^k + \sum_{i\in I} Z_i(\hat{x}^k)$ **then**
    $UB \leftarrow c^\top \hat{x}^k + \sum_{i\in I} Z_i(\hat{x}^k)$;
    $x^* \leftarrow \hat{x}^k$;
    $y^* \leftarrow (\hat{y}_1^k, \ldots, \hat{y}_{|I|}^k)$;
  end
  $k \leftarrow k + 1$;
end
return (x^*, y^*), UB, and LB;

The implementation of the cooling system over the planning horizon is staged over multiple phases, with each phase expanding over one or several years. Additional equipment can be installed only at the beginning of a phase. The decision-making process therefore involves determining both the total number of units to be installed and the timing of their installation across the different phases of the planning horizon. Figure 3 shows the structure of the district cooling system.

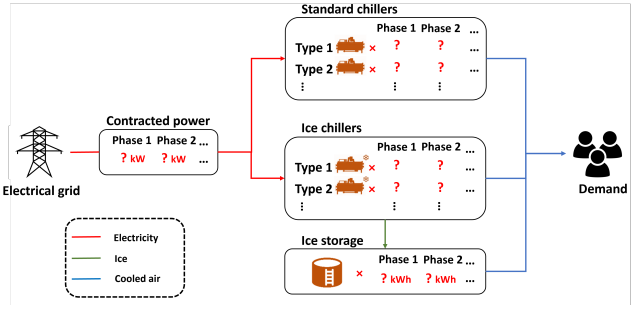


Figure 3: Optimal design problem of a district cooling system

Long-term decisions must account for both design costs and the subsequent impact on daily cooling system operations. Since modeling hourly chiller on/off status over the entire horizon is computationally intractable, we assume a yearly periodic cooling demand within each phase, with the overall demand level increasing across phases. We therefore employ the common practice of using a set of typical days to represent diverse operating conditions within each phase and build a detailed operational plan for each day.

The detailed MILP model is provided in Appendix B. Furthermore, in our numerical study, we considered 30 instances involving $\Phi = 3$ design phases and $|\mathcal{P}| = 5$ types of chillers. These instances vary according to the relative importance of the second-level operational cost in the objective function (represented by a factor $OPRate \in \{0.7, 1, 1.3\}$) and the number of typical days per phase (selected from {2,10,18,26,34,42,50,58,66,70}). The size of the MILP formulation depends, among other factors, on the number of typical days per phase. Further details are provided in Table 5 of Appendix B.

All tests are conducted on a computer with an Intel(R) Core(TM) i9-12900H (12th Gen) processor and 16 GB of RAM. The model is formulated in Python and solved using IBM ILOG CPLEX 22.1.1.0. During the solution process, the solver memory is limited to 30% of the total RAM, corresponding to approximately 4.8 GB.

6.2 Problem decomposition

The problem exhibits a two-level structure: the upper-level master problem determines the system design for each deployment phase, while the lower-level

subproblem optimizes the operational schedule for each typical day within these phases. Moreover, given a fixed system deployment plan, the lower-level problem further decomposes into a set of independent single-phase sub-problems, each one focusing on building the operational schedule for the typical days related to the considered phase.

In practice, both ice storage capacity and contracted power are determined in discrete multiples of basic units. Consequently, we model these decisions using integer variables representing the number of basic units of ice storage or contracted power to be determined in each phase. The upper-level master problem is thus a pure integer program, which enables us to use the generalized BD approach based on indicator Benders cuts described in Section 5.3.

Let $\phi \in \{1, \ldots, \Phi\}$ denote the index of a deployment phase and $p \in \mathcal{P}$ the index of a standard or ice chiller type. We introduce the following decision variables. $x_{\phi,p}$ denotes the number of chiller units of type p installed at the beginning of phase ϕ. Similarly, x^S_ϕ and x^C_ϕ represent the number of installed ice storage units and contracted power units in Phase ϕ, respectively. The vector of upper-level variables x is thus defined as:

$$x = (x_{1,1}, x_{2,1}, \cdots, x_{\Phi,1}, x_{1,2}, \cdots, x_{\Phi,|\mathcal{P}|}, x^S_1, \cdots, x^S_\Phi, x^C_1, \cdots, x^C_\Phi) \tag{72}$$

Let A denote the coefficient matrix of the coupling constraints. As can be seen from the detailed MILP formulation provided in Appendix B, all coefficients of A are non-negative so that there cannot be implicit equalities in the system $Ax \leq 0$. Consequently, the polar cone $K^o(A)$ is pointed.

6.3 Single-cut implementation of the algorithm

We first consider a single-cut implementation of our generalized BD algorithm, in which the operational scheduling sub-problem is considered as a whole and at most a single indicator Benders cut is added into the master problem at each iteration.

Applying Algorithm 4 provided in Appendix A to matrix A yields an effective equivalent matrix E^r, which takes the following block diagonal form:

$$E^r = \begin{pmatrix} \Lambda & 0 & \cdots & 0 & 0 \\ 0 & \Lambda & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \Lambda & 0 \\ 0 & 0 & \cdots & 0 & I^\Phi \end{pmatrix} \tag{73}$$

Here, I^Φ is a $\Phi \times \Phi$ identity matrix, and Λ is the following lower triangular matrix of size $\Phi \times \Phi$:

$$\Lambda = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \tag{74}$$

The diagonal structure of E^r is composed of $|\mathcal{P}| + 1$ blocks of the square matrix Λ, and of a single identity matrix I^Φ. For $p = 1...|\mathcal{P}|$, diagonal block p corresponds to the coupling constraints relative to chiller type p. Block $|\mathcal{P}| + 1$ corresponds to the ice storage capacity coupling constraints. Finally, block matrix I^Φ represent the constraints relative to the contracted power with the grid. Overall, matrix E^r is thus a square matrix with $\Phi \cdot (|\mathcal{P}| + 2)$ rows and columns. Furthermore, for $p = 1...|\mathcal{P}| + 1$, index $\Phi(p-1) + \phi$ of vector $E^r x$ computes the total number of units of type p or of storage capacity available at the beginning of phase ϕ while for $p = |\mathcal{P}| + 2$, index $\Phi(p-1) + \phi$ of vector $E^r x$ gives the number of base power units contracted for phase ϕ.

To assess the quality of the indicator cuts introduced in Section 3, we first solve each instance using a monolithic approach by providing the complete problem formulation directly to the CPLEX solver. This serves as a benchmark for our subsequent experiments. While all considered cuts are strong and guarantee the finite convergence of the algorithm, their effectiveness varies, potentially impacting the convergence rate. Specifically, we examine the following types of cuts:

1. **Classical no-good cut (CNG)**: Generate a classical no-good cut based on vector x describing the overall deployment plan. This cut ensures that if $x = \hat{x}$, then $Z(x) \geq Z(\hat{x})$.

2. **Strengthened no-good cut (SNG)**: Generate a no-good cut based on the value of vector $E^r x$. This vector describes the system state at the beginning of each deployment phase resulting from the accumulation of units installed up to the current phase. This cut ensures that if $E^r x = E^r \hat{x}$, then $Z(x) \geq Z(\hat{x})$. It is derived from the effectiveness theory presented in Section 3.6.

3. **Monotone indicator cut (MI)**: Generate a monotone indicator cut using the method introduced by [Liu et al., 2024]. This cut leverages the fact that matrix A has only non-negative entries and ensures that if $x \leq \hat{x}$, then $Z(x) \geq Z(\hat{x})$.

4. **Maximal indicator cut (MAI)**: Generate a maximal indicator cut following the approach introduced in Section 3 while using the effective equivalent matrix E^r. This cut ensures that if $E^r x \leq E^r \hat{x}$, then $Z(x) \geq Z(\hat{x})$.

For both the MAI and MI cuts, we also numerically compare the two alternative formulations introduced in Section 4, based respectively on binary expansion and unary expansion.

The numerical results are displayed in Tables 1 and 2. We report, for each instance and each solution approach, 'CalcTime' the total computation time, 'RelGap' the relative gap between the best known upper and lower bounds, and 'Iter' the number of iterations performed by the generalized BD algorithm.

First, we observe that CPLEX fails to solve any instance to optimality, as all runs terminate due to memory exhaustion. In contrast, the implementation of Algorithm 1 with CNG cuts significantly improves numerical efficiency, enabling

the optimal solution of 21 out of 30 instances within the allotted time. The results obtained using SNG cuts instead of CNG cuts are nearly identical in terms of both computational time and iteration count. This is expected, as SNG and CNG are effectively equivalent in the aggregated setting, with $lin(E^r) = lin(I^{\Phi\cdot(|\mathcal{P}|+2)}) = \mathbb{R}^{\Phi\cdot(|\mathcal{P}|+2)}$.

Second, the results presented in Tables 1 and 2 allow for a comparison between the two alternative reformulations of the MAI and MI cuts. In both cases, the unary expansion clearly outperforms the binary one. Specifically, when generating MI cuts, 20 out of 30 instances are solved to optimality using the unary formulation, compared to only 10 using the binary formulation. Similarly, when generating MAI cuts, 21 out of 30 instances are solved to optimality using the unary formulation, compared to only 17 using the binary formulation. This improvement may be due to the fact that, in our setting, each integer upper-level variable has a small domain, so the number of auxiliary binary variables introduced a priori into the master problem remains limited.

Finally, we compare the performance of MAI and MI cuts within Algorithm 1, using the unary expansion reformulation. For the 10 instances with OPRate = 0.7, the results are nearly identical in terms of computation time and iteration count. However, as OPRate increases, MAI cuts clearly outperform MI cuts. Among the 10 instances solved to optimality by both approaches, the average number of iterations decreases from 772.1 with MI cuts to 250.3 with MAI cuts. The average computation time also decreases from 4881s with MI cuts to 1102s with MAI cuts. For the remaining 10 instances, the average relative gap decreases from 0.11% with MI cuts to 0.06% with MAI cuts.

Table 1: Results of the single-cut implementation of the generalized Benders algorithm (Part 1/2)

		CPLEX		MAI unary			MAI binary			SNG		
OPRate	**Days**	**CalcTime**	**RelGap**	**CalcTime**	**RelGap**	**Iter**	**CalcTime**	**RelGap**	**Iter**	**CalcTime**	**RelGap**	**Iter**
0.7	2	m.	0.09%	194.91	opt	293	t.	0.03%	213	170.32	opt	295
	10	m.	0.12%	194.26	opt	208	568.52	opt	208	184.44	opt	209
	18	m.	0.12%	320.99	opt	228	728.06	opt	228	286.58	opt	229
	26	m.	0.55%	400.48	opt	221	714.56	opt	221	371.05	opt	222
	34	m.	0.26%	500.06	opt	235	771.15	opt	235	441.26	opt	236
	42	m.	46.26%	587.05	opt	230	833.65	opt	230	510.56	opt	231
	50	m.	39.89%	703.53	opt	245	1344.36	opt	245	626.73	opt	246
	58	m.	39.48%	800.80	opt	247	1023.42	opt	247	722.27	opt	248
	66	m.	39.54%	901.43	opt	253	1113.72	opt	253	802.45	opt	254
	70	m.	66.49%	844.18	opt	253	1143.00	opt	253	782.84	opt	254
1	2	m.	0.13%	406.45	opt	277	2532.82	opt	277	1265.09	opt	720
	10	m.	0.12%	1103.92	opt	399	t.	0.01%	327	4222.37	opt	1268
	18	m.	0.13%	781.58	opt	317	3191.84	opt	317	3418.84	opt	1058
	26	m.	0.50%	879.58	opt	274	1668.00	opt	274	3101.40	opt	873
	34	m.	0.54%	980.58	opt	280	1472.49	opt	280	3233.84	opt	859
	42	m.	37.69%	1181.74	opt	287	1870.41	opt	287	4083.74	opt	908
	50	m.	192.92%	1330.35	opt	295	2067.48	opt	295	4452.28	opt	920
	58	m.	33.46%	1483.18	opt	313	3286.45	opt	313	4978.99	opt	987
	66	m.	33.32%	1607.84	opt	305	t.	0.02%	234	4792.88	opt	937
	70	m.	196.03%	1653.84	opt	305	t.	0.02%	228	5067.01	opt	934
1.3	2	m.	0.02%	935.22	opt	451	5435.16	opt	451	6673.11	opt	1728
	10	m.	0.13%	t.	0.03%	1233	t.	0.06%	466	t.	0.07%	1246
	18	m.	0.47%	t.	0.02%	959	t.	0.04%	485	t.	0.05%	973
	26	m.	29.33%	t.	0.03%	683	t.	0.05%	447	t.	0.06%	809
	34	m.	1.05%	t.	0.06%	628	t.	0.07%	437	t.	0.09%	693
	42	m.	32.19%	t.	0.08%	553	t.	0.09%	405	t.	0.26%	637
	50	m.	29.73%	t.	0.08%	494	t.	0.09%	380	t.	0.26%	583
	58	m.	29.79%	t.	0.10%	461	t.	0.10%	412	t.	0.13%	515
	66	m.	177.89%	t.	0.08%	382	t.	0.08%	360	t.	0.10%	480
	70	m.	174.01%	t.	0.09%	364	t.	0.09%	346	t.	0.12%	435

m. = out of memory; t. = out of time limit; opt = optimality.

Table 2: Results of the single-cut implementation of the generalized Benders algorithm (Part 2/2)

		MI unary			MI binary			CNG		
OPRate	**Days**	**CalcTime**	**RelGap**	**Iter**	**CalcTime**	**RelGap**	**Iter**	**CalcTime**	**RelGap**	**Iter**
0.7	2	226.87	opt	295	1360.16	opt	295	166.32	opt	295
	10	213.34	opt	209	393.70	opt	209	190.35	opt	209
	18	383.70	opt	229	508.14	opt	229	281.55	opt	229
	26	427.09	opt	222	570.50	opt	222	345.25	opt	222
	34	2265.57	opt	236	647.01	opt	236	421.29	opt	236
	42	2542.55	opt	231	723.64	opt	231	527.44	opt	231
	50	732.17	opt	246	852.23	opt	246	633.08	opt	246
	58	809.33	opt	248	936.04	opt	248	730.92	opt	248
	66	854.85	opt	254	1028.73	opt	254	863.57	opt	254
	70	843.53	opt	254	1027.96	opt	254	806.80	opt	254
1	2	1487.80	opt	573	t.	0.02%	421	1111.70	opt	720
	10	5518.62	opt	1003	t.	0.04%	374	4101.57	opt	1268
	18	3985.31	opt	826	t.	0.02%	514	2976.59	opt	1058
	26	3471.83	opt	683	t.	0.02%	451	2821.57	opt	873
	34	3981.52	opt	674	t.	0.02%	505	3065.75	opt	859
	42	6578.14	opt	710	t.	0.03%	434	3710.93	opt	908
	50	7134.76	opt	716	t.	0.05%	310	4162.86	opt	920
	58	t.	1.16e-3%	744	t.	0.05%	324	4908.99	opt	987
	66	5359.42	opt	730	t.	0.02%	480	4486.57	opt	937
	70	6056.38	opt	737	t.	0.02%	498	4743.62	opt	934
1.3	2	5232.68	opt	1069	t.	0.01%	559	6287.91	opt	1728
	10	t.	0.06%	942	t.	0.08%	544	t.	0.06%	1471
	18	t.	0.05%	755	t.	0.05%	541	t.	0.05%	1141
	26	t.	0.06%	600	t.	0.06%	502	t.	0.06%	979
	34	t.	0.09%	529	t.	0.09%	454	t.	0.09%	845
	42	t.	0.25%	490	t.	0.28%	164	t.	0.26%	776
	50	t.	0.26%	449	t.	0.29%	162	t.	0.26%	710
	58	t.	0.13%	461	t.	0.15%	159	t.	0.13%	611
	66	t.	0.09%	381	t.	0.09%	410	t.	0.09%	567
	70	t.	0.12%	355	t.	0.10%	394	t.	0.10%	524

m. = out of memory; t. = out of time limit; opt = optimality.

6.4 Multi-cut implementation of the algorithm

We now consider a multi-cut implementation of our generalized BD algorithm, in which the operational scheduling sub-problem is decomposed into a set of independent MILP problems, each one corresponding to a single phase ϕ. In this variant of the algorithm, at most Φ indicator Benders cuts are thus added into the master problem at each iteration.

Each subproblem $\mathbf{SP}_\phi$ optimizes the hourly operation of all typical days in Phase ϕ, given the system configuration at the beginning of this phase. Consequently, the coupling constraints in this subproblem involves only upper level variables related to phases up to ϕ. This reflects the fact that only units installed in earlier phases are available in the current phase, while those scheduled for later phases cannot be used. Let $Z_\phi(x)$ denote the operation cost of Phase ϕ as a function of the system deployment plan x. Although $Z_\phi(\cdot)$ depends solely on the design variables determined prior to Phase ϕ, we retain x as the argument of $Z_\phi(\cdot)$ for notational consistency.

Applying Algorithm 4 provided in the Appendix A to A_ϕ, the coupling coefficient matrix of subproblem $\mathbf{SP}_\phi$, yields the following effective equivalent matrix E^r_ϕ.

$$E_\phi^r = \begin{pmatrix} \mathbf{u}_\phi & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{u}_\phi & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathbf{u}_\phi & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{e}_\phi \end{pmatrix} \tag{75}$$

Here, $\mathbf{u}_\phi$ and $\mathbf{e}_\phi$ are two $1 \times \Phi$ row vectors. $\mathbf{u}_\phi$ is a vector of length Φ where the first ϕ elements are 1 and the remaining elements are 0. $\mathbf{e}_\phi$ is a vector of length Φ where only the ϕ-th element is 1, and all other elements are 0:

$$\mathbf{u}_\phi = (\underbrace{1, 1, \ldots, 1}_{\phi}, \underbrace{0, 0, \ldots, 0}_{\Phi - \phi})$$

$$\mathbf{e}_\phi = (0, \ldots, 0, \underbrace{1}_{\phi\text{-th element}}, 0, \ldots, 0)$$

Furthermore, for $p \in \{1, \cdots, |\mathcal{P}| + 1\}$, index p of vector $E_\phi^r x$ gives the total number of units of type p or of storage capacity available at the beginning of phase ϕ while index $p = |\mathcal{P}| + 2$ provides the number of base power units contracted for phase ϕ.

We use the same experimental setup as the one described in the previous section to numerically assess the computational efficiency of our indicator cuts in a multi-cut implementation setting. We thus execute Algorithm 1, varying the types of cuts implemented at lines 27 and 30. Specifically, we examine the following types of cuts for each subproblem $\mathbf{SP}_\phi$:

1. **Classical no-good cut:** If $x_\varphi = \hat{x}_\varphi$ for all $\varphi \leq \phi$, then $Z_\phi(x) \geq Z_\phi(\hat{x})$.
2. **Strengthened no-good cut:** If $E_\phi^r x = E_\phi^r \hat{x}$, then $Z_\phi(x) \geq Z_\phi(\hat{x})$.
3. **Monotone indicator cut:** If $x_\varphi \leq \hat{x}_\varphi$ for all $\varphi \leq \phi$, then $Z_\phi(x) \geq Z_\phi(\hat{x})$.
4. **Maximal indicator cut:** If $E_\phi^r x \leq E_\phi^r \hat{x}$, then $Z_\phi(x) \geq Z_\phi(\hat{x})$.

where x_ϕ is defined as the decision vector for Phase ϕ determining the newly installed units and the contracted power, i.e.:

$$x_\phi = (x_{\phi,1}, x_{\phi,2}, \cdots, x_{\phi,|\mathcal{P}|}, x_\phi^S, x_\phi^C) \tag{76}$$

Results are reported in Tables 3 and 4. In all cases, the multi-cut implementation outperforms its single-cut counterpart by solving more instances to optimality and reducing the relative gaps.

Despite the improvement brought by the multi-cut implementation, Algorithm 1 using MI or CNG cuts still fails on instances with high OPRate. This is due to the lower effectiveness of these cuts, which leads to slower convergence. In contrast, the MAI and SNG implementations solve all instances to optimality. While their performance is similar for low OPRate values, the MAI

implementation with binary expansion performs best as the OPRate increases. For the 20 instances with OPRate equal to 1 and 1.3, binary MAI solves the instances in 442s on average, compared with 597s for SNG and 471s for unary MAI. This improvement over SNG is mainly explained by the smaller number of iterations required by MAI cuts. In addition, over all 30 instances, binary MAI has a lower average computation time per iteration than unary MAI, namely 2.77s versus 2.93s. This lower per-iteration cost arises because the binary formulation introduces fewer auxiliary variables in the master problem during the early stages of the algorithm, which improves efficiency when the total number of iterations remains moderate.

In summary, the numerical results presented in Subsection 6.3 and 6.4 demonstrate that the multi-cut implementation of our algorithm using MAI cuts and a binary expansion-based reformulation, consistently outperforms the other considered approaches in terms of both convergence speed and overall computational efficiency. This performance is achieved by combining a lower number of iterations with a reduced per-iteration computational cost.

Table 3: Results of multi-indicator Benders cuts (Part 1/2)

		MAI unary			**MAI binary**			**SNG**		
OPRate	**Days**	**CalcTime**	**RelGap**	**Iter**	**CalcTime**	**RelGap**	**Iter**	**CalcTime**	**RelGap**	**Iter**
0.7	2	38.28	opt	72	38.97	opt	72	35.97	opt	72
	10	87.36	opt	105	87.07	opt	105	85.48	opt	105
	18	137.43	opt	124	135.13	opt	124	130.70	opt	124
	26	161.61	opt	121	164.42	opt	121	162.00	opt	121
	34	215.98	opt	137	194.90	opt	137	200.50	opt	137
	42	236.15	opt	131	227.56	opt	131	230.54	opt	131
	50	288.73	opt	140	283.35	opt	140	278.92	opt	140
	58	339.00	opt	143	324.65	opt	143	330.04	opt	143
	66	400.85	opt	150	380.37	opt	150	373.09	opt	150
	70	390.80	opt	150	381.49	opt	150	382.94	opt	150
1	2	57.75	opt	77	49.50	opt	77	63.51	opt	87
	10	157.80	opt	119	141.48	opt	119	185.73	opt	129
	18	240.34	opt	145	224.09	opt	145	287.62	opt	154
	26	251.02	opt	126	235.48	opt	126	315.16	opt	134
	34	351.48	opt	149	314.89	opt	149	438.75	opt	161
	42	396.66	opt	145	398.73	opt	145	526.20	opt	157
	50	479.28	opt	154	458.74	opt	154	635.72	opt	166
	58	574.10	opt	160	540.23	opt	160	732.65	opt	170
	66	647.88	opt	163	594.29	opt	163	801.42	opt	174
	70	642.11	opt	160	621.24	opt	160	875.07	opt	172
1.3	2	51.49	opt	68	60.81	opt	68	55.57	opt	73
	10	209.05	opt	115	222.02	opt	115	227.24	opt	123
	18	323.91	opt	133	305.87	opt	133	315.09	opt	136
	26	378.36	opt	124	357.01	opt	124	413.73	opt	130
	34	538.82	opt	143	495.14	opt	143	1268.77	opt	150
	42	638.36	opt	143	598.14	opt	143	650.89	opt	150
	50	699.48	opt	149	699.97	opt	149	760.07	opt	157
	58	824.49	opt	147	755.02	opt	147	912.29	opt	155
	66	929.81	opt	153	867.02	opt	153	983.64	opt	161
	70	1030.24	opt	150	905.18	opt	150	1495.98	opt	159

m. = out of memory; t. = out of time limit; opt = optimality.

7 Conclusion

We presented a novel approach for solving large-scale Benders-decomposable problems with a pure-integer master problem and MILP subproblems. By leveraging the monotonicity properties of subproblem value functions, this method

Table 4: Results of multi-indicator Benders cuts (Part 2/2)

OPRate	Days	MI unary			MI binary			CNG		
		CalcTime	RelGap	Iter	CalcTime	RelGap	Iter	CalcTime	RelGap	Iter
0.7	2	109.10	opt	123	166.04	opt	123	93.60	opt	123
	10	130.12	opt	122	139.05	opt	122	123.31	opt	122
	18	216.99	opt	142	219.73	opt	142	204.00	opt	142
	26	258.27	opt	138	260.03	opt	138	243.31	opt	138
	34	333.02	opt	155	337.69	opt	155	320.17	opt	155
	42	400.79	opt	149	393.79	opt	149	374.33	opt	149
	50	475.55	opt	158	470.03	opt	158	439.30	opt	158
	58	586.55	opt	163	554.64	opt	163	525.63	opt	163
	66	618.99	opt	168	620.53	opt	168	577.79	opt	168
	70	624.33	opt	167	610.46	opt	167	567.33	opt	167
1	2	288.96	opt	173	531.28	opt	173	250.12	opt	205
	10	628.53	opt	225	832.06	opt	225	603.62	opt	259
	18	659.19	opt	214	722.28	opt	214	636.19	opt	239
	26	703.69	opt	188	762.96	opt	188	890.16	opt	240
	34	766.80	opt	200	763.33	opt	200	846.00	opt	230
	42	991.25	opt	200	961.23	opt	200	1075.38	opt	231
	50	1117.62	opt	208	1127.41	opt	208	1282.09	opt	241
	58	1326.69	opt	218	1361.81	opt	218	1438.08	opt	247
	66	1347.30	opt	218	1334.35	opt	218	4748.22	opt	250
	70	1454.09	opt	215	1423.60	opt	215	t.	0.05%	179
1.3	2	3058.63	opt	662	t.	0.01%	437	t.	0.01%	675
	10	t.	0.05%	765	t.	0.07%	435	t.	0.10%	276
	18	t.	0.03%	615	t.	0.04%	410	t.	0.06%	223
	26	t.	0.04%	527	t.	0.05%	388	t.	0.07%	185
	34	t.	0.07%	490	t.	0.08%	381	t.	0.26%	180
	42	t.	0.25%	452	t.	0.26%	323	t.	0.27%	243
	50	t.	0.25%	413	t.	0.26%	339	t.	0.25%	510
	58	t.	0.11%	405	t.	0.09%	348	t.	0.11%	387
	66	t.	0.08%	354	t.	0.08%	328	t.	0.12%	297
	70	t.	0.12%	336	t.	0.13%	220	t.	0.11%	332

m. = out of memory; t. = out of time limit; opt = optimality.

extends classical Benders decomposition and enables the construction of non-convex indicator cuts.

We first introduced the monotonicity property of parametric MILP problems, which forms the basis for defining the indicator cuts. Then, we defined and developed a theoretical framework to evaluate the effectiveness of different indicator cuts, allowing for a reduction in the size of their formulations and enabling comparisons between different types of cuts. Finally, we proposed two disjunctive formulations for implementing indicator cuts within a MILP framework, along with algorithmic enhancements to improve the efficiency of the Benders decomposition process. A series of numerical experiments were conducted on a multi-phase design problem for a district cooling system. The results show that the proposed indicator Benders cut method outperforms direct solving with the commercial solver CPLEX in both solving time and memory required. The comparisons across different types of indicator cuts also validate the theoretical framework for cut effectiveness.

The future work involves two main directions. First, the trade-off between effectiveness and formulation complexity plays a crucial role in the overall computational performance. For a general coefficient matrix of upper-level variables in the coupling constraints, we aim to design a generic procedure for constructing indicator Benders cuts that appropriately balances this trade-off.

Second, a similar approach may be extended to the case where upper-level variables are mixed-integer. However, when upper-level variables are continuous, the indicator function is not lower semi-continuous, which prevents a direct

extension of the current framework. Therefore, a more general methodological approach needs to be developed.

Appendices

Appendix A Algorithms

This section describes the algorithm used during the pre-processing step of our generalized BD algorithm to build an effective equivalent to each coupling matrix A_i, $i \in I$.

A.1 Algorithm for finding the basis of the largest subspace contained in a cone

Given a matrix $A \in \mathbb{R}^{k \times m}$, we denote by $cone(A)$ the cone generated by its row vectors. For an arbitrary matrix A, it is possible that $cone(A)$ is not pointed, i.e., that it contains a subspace of $\mathbb{R}^m$. In this section, we explain how to compute a description of the lineality space L of $cone(A)$, that is, the inclusion-wise largest linear subspace contained in $cone(A)$. This description takes the form of a matrix $E^b \in \mathbb{R}^{l^= \times m}$ whose row vectors form an orthogonal basis of L. In particular, the linear space spanned by the row vectors of E^b, denoted $lin(E^b)$, coincides with L, and the row vectors of E^b are mutually orthogonal.

A vector v belongs to the lineality space L of $cone(A)$ only if both v and $-v$ belong to $cone(A)$. Furthermore, L is fully determined by the generating vectors of $cone(A)$: the matrix E^b can thus be obtained by considering only the row vectors of A and by determining the subset of these row vectors belonging to L. By definition of $cone(A)$, each row vector a_i, $i = 1...k$, of matrix A belongs to $cone(A)$. To determine whether $a_i \in L$, it is therefore sufficient to check whether $-a_i \in cone(A)$, which can be formulated as a conic feasibility problem over the generators of $cone(A)$.

Algorithm 2 performs this test for each row vector a_i of A by checking the feasibility of problem (77)-(78). If the problem is feasible, then $a_i \in cone(A) \cap (-cone(A))$, i.e., $a_i \in L$. All row vectors of A belonging to L are collected in V', and Gram-Schmidt orthogonalization is finally applied to obtain an orthogonal basis stored as the rows of E^b.

A.2 Algorithm for vector projection on the perpendicular space of a basis

In this section, we are given two matrices: a matrix E^b whose rows form a basis of a linear space, and a matrix $A \in \mathbb{R}^{k \times m}$, the coefficient matrix of the coupling constraints. We construct a matrix $\widetilde{A} \in \mathbb{R}^{k \times m}$ of the same size as A by projecting each row of A onto the orthogonal complement of $lin(E^b)$.

Algorithm 2: Orthonormal basis of the lineality space of cone(A)

Data: A matrix $A \in \mathbb{R}^{k \times m}$.
Result: A matrix $E^b \in \mathbb{R}^{k^= \times m}$ whose rows form an orthonormal basis of the lineality space of cone(A).

Let $a_r^\top$ be the r-th row of A, for all $r = 1, \ldots, k$;
Set $V' \leftarrow \emptyset$ and $i \leftarrow 1$;
while $i \leq k$ **do**
  Solve the feasibility problem with respect to λ:

$$a_i = -\sum_{j=1,\, j \neq i}^{k} \lambda_j a_j \tag{77}$$

$$\lambda_j \geq 0, \qquad \forall j \in \{1, \cdots, k\} \tag{78}$$

  if *(77)–(78) is feasible* **then**
    $V' \leftarrow V' \cup \{a_i\}$;
  end
  $i \leftarrow i + 1$;
end
if V' *is empty* **then**
  return 0;
end
Apply Gram–Schmidt orthogonalization to V' and obtain an orthonormal set $\{e_1, \ldots, e_{k^=}\}$;
Construct E^b by stacking $e_r^\top$ as rows for $r = 1, \ldots, k^=$;
return E^b;

More precisely, let $\tilde{a}_i^\top$ denote the i-th row of $\widetilde{A}$ and let $a_i^\top$ be the i-th row of A. Then, for $i = 1, \ldots, k$, we have

$$\tilde{a}_i^\top = proj_{lin(E^b)^\perp}\left(a_i^\top\right). \tag{79}$$

Algorithm 3 constructs $\widetilde{A}$ by removing, row by row, the component of A that lies in the linear space spanned by the rows of E^b. After computing an orthonormal basis q_r of $\mathrm{lin}(E^b)$ via Gram–Schmidt, each row $a_i^\top$ is decomposed into its projection onto $\mathrm{lin}(E^b)$ and its residual in $\mathrm{lin}(E^b)^\perp$; the latter residual is stored as $\tilde{a}_i^\top$. This projected matrix $\widetilde{A}$ will be used in subsequent constructions where only the directions orthogonal to $\mathrm{lin}(E^b)$ matter.

A.3 Algorithm for finding the extreme rays of a set of vectors

In this section, we are given a matrix $A \in \mathbb{R}^{k \times m}$ and denote by $cone(A)$ the cone generated by its row vectors. We assume that $cone(A)$ is pointed. Our

Algorithm 3: Projecting A onto $\text{lin}(E^b)^\perp$

Data: A matrix $E^b \in \mathbb{R}^{k^= \times m}$ whose rows are orthonormal, and a matrix $A \in \mathbb{R}^{k \times m}$.

Result: A matrix $\widetilde{A} \in \mathbb{R}^{k \times m}$ whose i-th row is the projection of the i-th row of A onto $\text{lin}(E^b)^\perp$.

if *E^b is the zero matrix (or $k^= = 0$)* **then**
  return A;
end
Let $e_r^\top$ be the r-th row of E^b, for all $r = 1, \ldots, k^=$;
Initialize $\widetilde{A} \leftarrow$ an empty matrix with k rows;
for $i \leftarrow 1$ **to** k **do**
  Let $a_i^\top$ be the i-th row of A;
  Set $\tilde{a}_i^\top \leftarrow a_i^\top$;
  for $r \leftarrow 1$ **to** $k^=$ **do**
    Set $\tilde{a}_i^\top \leftarrow \tilde{a}_i^\top - (a_i^\top \cdot e_r^\top) e_r^\top$;
  end
  Set the i-th row of $\widetilde{A}$ to be $\tilde{a}_i^\top$;
end
return $\widetilde{A}$;

goal is to construct a matrix E^r with the same number of columns as A whose rows are generators of the extreme rays of $cone(A)$.

Algorithm 4 constructs E^r by iteratively removing redundant generators from the current set of rows. For each candidate vector v_i, it tests the feasibility of (80)–(81). If the system is feasible, then v_i can be expressed as a conic combination of the remaining vectors and therefore cannot generate an extreme ray, hence it is removed. This elimination process is repeated until no further vector can be removed, and the remaining set of rows forms the output matrix E^r.

Appendix B Optimal design of a local cooling system: MILP formulation

This section provides a detailed MILP formulation for the multi-phase local cooling system design problem.

B.1 Notation

We first introduce the notation used to formulate the model. We adopt the notation rule of [Richarz et al., 2022] to denote variables using bold letters.

Algorithm 4: Identifying extreme rays of a cone generated by a set of vectors

Data: A matrix $A \in \mathbb{R}^{k \times m}$ such that the cone $\{x \in \mathbb{R}^m \mid Ax \leq 0\}$ has nonempty interior.

Result: A matrix $E^r \in \mathbb{R}^{k^< \times m}$ whose rows are the extreme rays of $cone(A)$.

Let $v_r^\top$ be the r-th row of A, for all $r = 1, \ldots, k$;
Set $I \leftarrow \{1, \cdots, k\}, V \leftarrow \{v_1, \cdots, v_k\}$;
Set `changed` $\leftarrow$ `True`;
while *changed* **do**
  Set `changed` $\leftarrow$ `False`;
  for $i \in I$ **do**
    Solve the system:

$$v_i = \sum_{j \in I \setminus \{i\}} \lambda_j v_j \tag{80}$$

$$\lambda_j \geq 0, \qquad \forall j \in I \tag{81}$$

    if *The system (80 - 81) is feasible* **then**
      $V \leftarrow V \setminus \{v_i\}$; `#Remove` v_i `from` V
      $I \leftarrow I \setminus \{i\}$; `#Remove` i `from` I
      `changed` $\leftarrow$ `True`;
      break;
    end
  end
end
Construct E^r by stacking vectors in V as rows;
return E^r;

B.1.1 Indices

- ϕ: design phase.
- d: representative day.
- $h \in \{0 \cdots, H-1\}$: time-step in a day.
- p: type of chillers.
- $c \in \{COLD, ICE\}$: commodity produced by chillers.
- b: breakpoint of the piecewise linear (PWL) approximation of the performance curve of a chiller.

B.1.2 Sets

- $\mathcal{D}_\phi$: set of representative days chosen for phase ϕ.

- $\mathcal{D} = \{(\phi, d) | \phi \in \{1, ..., \Phi\}, d \in \mathcal{D}_\phi\}$: set of all phase-day pairs.
- $\mathcal{P} = \{STDC1, STDC2, STDC3, ICEC1, ICEC2\}$: set of all chiller types.
- $\mathcal{P}^S = \{STDC1, STDC2, STDC3\}$: set of all standard chillers types.
- $\mathcal{P}^I = \{ICEC1, ICEC2\}$: set of all ice chiller types.
- $\mathcal{C}_p \subseteq \{COLD, ICE\}$: set of commodities that chillers of category p can produce

B.1.3 Parameters

- Φ: number of design phases
- Y: number of years of the evaluation horizon.
- $\underline{y}_\phi$, $\overline{y}_\phi$: the first and the last year belonging to phase ϕ.
- $w_{\phi,d}$: weight of the representative day (ϕ, d).
- H: number of time-steps within a representative day.
- Δt: duration of a time-step.
- $Dem_{\phi,d,h}$: cooling demand at period (ϕ, d, h).
- $EP_{\phi,d,h,c}$: electricity price for producing commodity c at period (ϕ, d, h).
- γ: annual discount rate.
- α_ϕ: design actualization rate at the beginning of a phase ϕ.
- β_ϕ: operation actualization rate during phase ϕ.
- $P^{min}_{p,c}$, $P^{max}_{p,c}$: minimum and maximum output power of a chiller of type p producing commodity $c \in \mathcal{C}_p$.
- $B_{\phi,d,h,p,c}$: number of breakpoints of the PWL approximated performance curve of a chiller of type p producing commodity c at time period (ϕ, d, h). This performance curve gives the amount of electricity consumed as a function of the quantity of commodity c produced.
- $a_{\phi,d,h,p,c,b}$, $o_{\phi,d,h,p,c,b}$: abscissa and ordinate of breakpoint $b \in \{1, ..., B_{\phi,d,h,p,c}\}$ in PWL approximation.
- LT_p: lifetime of the chillers of type p
- FC^P_p, FC^L_p: purchase cost and labor cost of building a chiller of type p.
- FC^M_p: annual maintenance cost of a chiller of type p.

- $FC^S_{\phi,p}$: salvage value at the end of the evaluation horizon of a chiller of type p installed in phase ϕ.
- $FC_{\phi,p}$: total design cost of installing a chiller of type p in phase ϕ.
- LC_ϕ: total design cost of building one basic unit of ice storage capacity in phase ϕ.
- SC: subscription cost of a basic unit of the maximum instantaneous power requested from the grid.
- SD^{max}_p: maximum allowed number of chillers of type p.
- $StoC^{max}$: maximum allowed ice storage capacity.
- C^{max}: maximum annual contracted power.
- $OPRate$: weighting factor of the operational cost in the objective function.

The net present value of the future cost is calculated using an annual discount rate denoted by γ. We use two distinct actualization rates in our problem formulation. The design actualization rate, $\alpha_\phi = \frac{1}{(1+\gamma)^{\underline{y}_\phi - 1}}$, is used to compute the net present value of the system design cost at the beginning of a phase ϕ. The operation actualization rate, $\beta_\phi = \sum_{y=\underline{y}_\phi}^{\overline{y}_\phi} \frac{1}{(1+\gamma)^y}$, is used to calculate the net present value of the operation cost during phase ϕ.

The salvage value $FC^S_{\phi,p}$ is assumed to be proportional to the residual lifetime $LT_p - (Y - \underline{y}_\phi + 1)$ of the chiller, i.e. $FC^S_{\phi,p} = FC^P_p(\frac{LT_p-(Y-\underline{y}_\phi+1)}{LT_p})\frac{1}{(1+\gamma)^Y}$. The total installation cost LC_ϕ of building one unit of ice storage capacity in phase ϕ is obtained in the same way. The total design cost of installing a chiller of type p in phase ϕ, $FC_{\phi,p}$, is computed as follows:

$$FC_{\phi,p} = \alpha_\phi(FC^P_p + FC^L_p) + \sum_{\varphi=\phi}^{\Phi} \beta_\varphi FC^M_p - FC^S_{\phi,p} \tag{82}$$

B.2 Variables

B.2.1 Design variables

Design variables represent long-term design decisions describing the structure of the system at each phase of the deployment plan.

- $\mathbf{SD}_{\phi,p}$: (integer) number of chillers of type p to be built at the beginning of phase ϕ.
- $\mathbf{SD}^S_\phi$: (integer) number of basic units of storage capacity to be built at the beginning of phase ϕ.
- $\mathbf{SD}^C_\phi$: (integer) number of basic units of contracted power to be built at the beginning of phase ϕ.

Therefore, the available units of type p for system operation in phase ϕ are the accumulated numbers of chillers $\sum_{\varphi=1}^{\phi} \mathbf{SD}_{\varphi,p}$ and the accumulated ice storage capacity $\sum_{\varphi=1}^{\phi} Sto_{step}\mathbf{SD}_{\varphi}^{S}$, and the contracted power as the maximum instantaneous electricity consumption power is $C_{step}\mathbf{SD}_{\phi}^{C}$.

B.2.2 Operation variables

Operation variables describe the operation scheduling to be built for each selected day $d \in \mathcal{D}_\phi$ of each phase ϕ.

- $\mathbf{S}_{\phi,d,h,p,c}$: (integer) number of active chillers of type p producing commodity c during time period ϕ, d, h.
- $\mathbf{P}_{\phi,d,h,p,c}$: (continuous) total amount of commodity c produced by active chillers of type p during time period ϕ, d, h.
- $\mathbf{Q}_{\phi,d,h,p,c}$: (continuous) total electric energy consumed by active chillers of type p to produce commodity c during time period ϕ, d, h.
- $\mathbf{STO}_{\phi,d,h}$: (continuous) ice storage level in the tank at the beginning of ϕ, d, h.
- $\mathbf{R}_{\phi,d,h}$: (continuous) amount of energy released from the ice storage during ϕ, d, h.

B.3 Objective function

The objective function (83) aims at minimizing the sum of the actualized design and operation costs of the system.

$$\min \sum_{\phi=1}^{\Phi} \left[\left(\sum_{p\in\mathcal{P}} FC_{\phi,p}\mathbf{SD}_{\phi,p} + \ LC_{\phi}\mathbf{SD}_{\phi}^{S} + \alpha_{\phi} SC\mathbf{SD}_{\phi}^{C} \right) + OPRate \cdot \beta_{\phi} \sum_{d\in\mathcal{D}_\phi} w_{\phi,d} \sum_{h=0}^{H-1} \sum_{p\in\mathcal{P}} \sum_{c\in\mathcal{C}_p} EP_{\phi,d,h,c}\mathbf{Q}_{\phi,d,h,p,c} \right] \quad (83)$$

B.4 Constraints

B.4.1 Design Constraints

Design constraints involve only design variables.

For each chiller type $p \in \mathcal{P}$, the total number of installed chillers should be less than the maximum allowed number.

$$\sum_{\varphi=1}^{\Phi} \mathbf{SD}_{\varphi,\mathbf{p}} \leq SD_{p}^{max} \quad (84)$$

Similarly, the total ice storage capacity should stay below the maximum allowed value.

$$\sum_{\varphi=1}^{\Phi} StoC_{step}\mathbf{SD}_{\varphi}^{S} \leq StoC^{max} \tag{85}$$

Finally, for each phase ϕ, the maximum allowed instantaneous power contracted with the grid should stay below the predefined upper bound.

$$C_{step}\mathbf{SD}_{\phi}^{C} \leq C^{max} \tag{86}$$

B.4.2 Coupling Constraints

Coupling constraints link the design variables and operation variables. For each chiller type $p \in \mathcal{P}$ and each time period (ϕ, d, h), the total number of active chillers should not exceed the number of chillers currently installed in the system:

$$\sum_{c \in \mathcal{C}_p} \mathbf{S}_{\phi,d,h,p,c} \leq \sum_{\varphi=1}^{\phi} \mathbf{SD}_{\varphi,p} \tag{87}$$

Similarly, the amount of ice stored in the tank at the end of each time period (ϕ, d, h) should stay below the current storage capacity.

$$\mathbf{STO}_{\phi,d,h} \leq \sum_{\varphi=1}^{\phi} StoC_{step}\mathbf{SD}_{\varphi}^{S} \tag{88}$$

Finally, in each time period (ϕ, d, h) the total electricity consumption of all chillers should not exceed $\mathbf{C}_{\phi}\Delta t$, the contracted power times the duration of a period.

$$\sum_{p \in \mathcal{P}} \sum_{c \in \mathcal{C}_p} \mathbf{Q}_{\phi,d,h,p,c} \leq C_{step}\mathbf{SD}_{\phi}^{C}\Delta t \tag{89}$$

B.4.3 Operation Constraints

Constraints including only operation variables are classified as operation constraints. We have the following set of constraints for each period $\phi, d, h = (\phi, d, h)$.

Demand satisfaction The cooling demand must be satisfied at all time. This can be done by directly producing chilled water with the chillers and/or by melting some ice from the thermal storage tank.

$$\sum_{p \in \mathcal{P}} \mathbf{P}_{\phi,d,h,p,COLD} + \mathbf{R}_{\phi,d,h} = Dem_{\phi,d,h} \tag{90}$$

Chillers For each chiller type $p \in \mathcal{P}$ and each commodity $c \in \mathcal{C}_p$, the total amount of produced commodity should lie within the allowed production range:

$$P_{p,c}^{min}\mathbf{S}_{\phi,d,h,p,c} \leq \mathbf{P}_{\phi,d,h,p,c} \leq P_{p,c}^{max}\mathbf{S}_{\phi,d,h,p,c} \tag{91}$$

Electric consumption In our case, all the chillers' performance curves are convex so that the PWL approximations are also convex. Since this problem naturally aims at minimizing the electricity consumption given output cooling demand, having convex performance curves would allow us to formulate the PWL approximations without binary auxiliary variables. For a set of identical active chillers producing the same commodity, the optimal load allocation is that they have the same output power. This property can be exploited to compute the electric consumption of the chillers in a given time period in an aggregated way. The total electric consumption of a set of $\mathbf{S}_{\phi,d,h,p,c}$ active chillers of a given type $p \in \mathcal{P}$ producing globally an amount $\mathbf{P}_{\phi,d,h,p,c}$ of commodity $c \in \mathcal{C}_p$ in time period ϕ, d, h is thus given by an aggregate performance curve. This one takes the form of a piece-wise linear function displaying $B_{t,p,c}$ breakpoints, whose coordinates are $(\mathbf{S}_{\phi,d,h,p,c}a_{\phi,d,h,p,c,b}, \mathbf{S}_{\phi,d,h,p,c}o_{\phi,d,h,p,c,b})$. Therefore, for each $p \in \mathcal{P}$, each $c \in \mathcal{C}_p$ and each $b \in \{1...B_{\phi,d,h,p,c} - 1\}$, we have the following inequality:

$$\mathbf{Q}_{\phi,d,h,p,c} \geq s_{\phi,d,h,p,c,b}\mathbf{P}_{\phi,d,h,p,c} + c_{\phi,d,h,p,c,b}\mathbf{S}_{\phi,d,h,p,c} \tag{92}$$

Here, $s_{\phi,d,h,p,c,b} = \frac{o_{\phi,d,h,p,c,b+1} - o_{\phi,d,h,p,c,b}}{a_{\phi,d,h,p,c,b+1} - a_{\phi,d,h,p,c,b}}$ is the slope and $c_{\phi,d,h,p,c,b} = o_{\phi,d,h,p,c,b} - s_{\phi,d,h,p,c,b}a_{\phi,d,h,p,c,b}$ the constant value of the line segment between breakpoints b and $b+1$ of the aggregate performance curve. Note that Constraints (92) accurately compute the electric consumption of a set of chillers only if the aggregate performance curve is convex.

Ice storage The amount of ice released during ϕ, d, h should not exceed the amount stored at the beginning of ϕ, d, h.

$$\mathbf{R}_{\phi,d,h} \leq \mathbf{STO}_{\phi,d,h} \tag{93}$$

The time evolution of the ice inventory should comply with inventory balance equations. For periods (ϕ, d, h) such that $h \in \{0...H-2\}$, we have:

$$\mathbf{STO}_{\phi,d,h} + \sum_{p \in \mathcal{P}^I} \mathbf{P}_{\phi,d,h,ICEC,p,ICE} - \mathbf{R}_{\phi,d,h} = \mathbf{STO}_{\phi,d,h+1} \tag{94}$$

Constraints (94) state that the amount of ice stored at the beginning of hour $h+1$, $\mathbf{STO}_{\phi,d,h+1}$, is equal to the amount of ice already stored at the beginning of hour h, $\mathbf{STO}_{\phi,d,h}$, plus the total amount of ice produced by the ice chillers during h minus the ice melted during h.

Moreover, as done in [Wakui et al., 2019], we assume that the selected day d will be cyclically repeated $w_{\phi,d}$ times each year and impose the cyclic condition $\mathbf{STO}_{\phi,d,H} = \mathbf{STO}_{\phi,d,0}$ for the ending inventory of time step $H-1$ of each selected day. This gives:

$$\mathbf{STO}_{\phi,d,H-1} + \sum_{p \in \mathcal{P}^I} \mathbf{P}_{\phi,d,H-1,ICEC,p,ICE} - \mathbf{R}_{\phi,d,H-1} = \mathbf{STO}_{\phi,d,0} \tag{95}$$

B.5 Instances and problem size

The instances involve $\Phi = 3$ phases, and $|\mathcal{P}| = 5$ chiller types. The numerical value of all input data were set based on this case study, except for the number of operational days which was varied between 2 and 70.

The size of the MILP formulation depends on the number of typical days selected in each phase. It is reported in Table 5. The column Var. gives the total number of variables in the complete model, while Int. Var. reports the number of discrete variables, including both binary and general integer variables. Constr. denotes the total number of constraints, and Coup. Constr. denotes the number of coupling constraints linking the design decisions and the operational variables.

Table 5: Model size with respect to the number of typical days

Days	Var.	Int. Var.	Constr.	Coup. Constr.
2	15147	3909	24204	2189
10	35307	9093	56460	5069
18	55467	14277	88716	7949
26	74787	19245	119628	10709
34	95787	24645	153228	13709
42	115947	29829	185484	16589
50	136107	35013	217740	19469
58	156267	40197	249996	22349
66	176427	45381	282252	25229
70	186507	47973	298380	26669

References

[Ahmed et al., 2022] Ahmed, S., Cabral, F. G., and Freitas Paulo da Costa, B. (2022). Stochastic Lipschitz dynamic programming. *Mathematical Programming*, 191(2):755–793.

[Angulo et al., 2016] Angulo, G., Ahmed, S., and Dey, S. S. (2016). Improving the integer L-shaped method. *INFORMS Journal on Computing*, 28(3):483–499.

[Balas, 2018] Balas, E. (2018). *Disjunctive programming*. Springer.

[Benders, 1962] Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

[Bolusani and Ralphs, 2022] Bolusani, S. and Ralphs, T. K. (2022). A framework for generalized Benders' decomposition and its application to multilevel optimization. *Mathematical Programming*, 196:389–426.

[Chen and Luedtke, 2022] Chen, R. and Luedtke, J. (2022). On generating Lagrangian cuts for two-stage stochastic integer programs. *INFORMS Journal on Computing*, 34(4):2332–2349.

[Codato and Fischetti, 2006] Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

[Fazel-Zarandi and Beck, 2012] Fazel-Zarandi, M. M. and Beck, J. C. (2012). Using logic-based benders decomposition to solve the capacity-and distance-constrained plant location problem. *INFORMS Journal on Computing*, 24(3):387–398.

[Forbes et al., 2024] Forbes, M., Harris, M., Jansen, H., van der Schoot, F., and Taimre, T. (2024). Combining optimisation and simulation using logic-based Benders decomposition. *European Journal of Operational Research*, 312(3):840–854.

[Füllner and Rebennack, 2022] Füllner, C. and Rebennack, S. (2022). Non-convex nested Benders decomposition. *Mathematical Programming*, 196(1-2):987–1024.

[Gade et al., 2014] Gade, D., Küçükyavuz, S., and Sen, S. (2014). Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64.

[Gendron et al., 2014] Gendron, B., Lucena, A., da Cunha, A. S., and Simonetti, L. (2014). Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26(4):645–657.

[Geoffrion, 1972] Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260.

[Hooker, 2011] Hooker, J. (2011). *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley & Sons.

[Hooker and Ottosson, 2003] Hooker, J. and Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming Series A*, 96:33–60.

[Hooker, 2005] Hooker, J. N. (2005). A hybrid method for the planning and scheduling. *Constraints*, 10(4):385–401.

[Hooker, 2007] Hooker, J. N. (2007). Planning and scheduling by logic-based benders decomposition. *Operations research*, 55(3):588–602.

[Hooker, 2019] Hooker, J. N. (2019). Logic-based Benders decomposition for large-scale optimization. In *Large Scale Optimization in Supply Chains and Smart Manufacturing*, pages 1–26. Springer.

[Laporte and Louveaux, 1993] Laporte, G. and Louveaux, F. V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research letters*, 13(3):133–142.

[Li and Grossmann, 2018] Li, C. and Grossmann, I. E. (2018). An improved L-shaped method for two-stage convex 0–1 mixed integer nonlinear stochastic programs. *Computers & Chemical Engineering*, 112:165–179.

[Liu et al., 2024] Liu, B., Bissuel, C., Courtot, F., Gicquel, C., and Quadri, D. (2024). A generalized benders decomposition approach for the optimal design of a local multi-energy system. *European Journal of Operational Research*, 318(1):43–54.

[Martínez et al., 2022] Martínez, K. P., Adulyasak, Y., and Jans, R. (2022). Logic-based benders decomposition for integrated process configuration and production planning problems. *INFORMS Journal on Computing*, 34(4):2177–2191.

[Ntaimo, 2010] Ntaimo, L. (2010). Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations Research*, 58(1):229–243.

[Ntaimo, 2013] Ntaimo, L. (2013). Fenchel decomposition for stochastic mixed-integer programming. *Journal of Global Optimization*, 55:193–235.

[Owen and Mehrotra, 2002] Owen, J. H. and Mehrotra, S. (2002). On the value of binary expansions for general mixed-integer linear programs. *Operations Research*, 50(5):810–819.

[Papadakos, 2008] Papadakos, N. (2008). Practical enhancements to the magnanti–wong method. *Operations Research Letters*, 36(4):444–449.

[Philpott et al., 2020] Philpott, A. B., Wahid, F., and Bonnans, J. F. (2020). MIDAS: A mixed integer dynamic approximation scheme. *Mathematical Programming*, 181(1):19–50.

[Qi and Sen, 2017] Qi, Y. and Sen, S. (2017). The ancestral Benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Mathematical Programming*, 161:193–235.

[Rahmaniani et al., 2020] Rahmaniani, R., Ahmed, S., Crainic, T. G., Gendreau, M., and Rei, W. (2020). The Benders dual decomposition method. *Operations Research*, 68(3):878–895.

[Richarz et al., 2022] Richarz, J., Henn, S., Osterhage, T., and Müller, D. (2022). Optimal scheduling of modernization measures for typical non-residential buildings. *Energy*, 238:121871.

[Sen and Higle, 2005] Sen, S. and Higle, J. L. (2005). The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20.

[Sen and Sherali, 2006] Sen, S. and Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223.

[van der Laan and Romeijnders, 2023] van der Laan, N. and Romeijnders, W. (2023). A converging Benders' decomposition algorithm for two-stage mixed-integer recourse models. *Operations Research*. in press.

[Wakui et al., 2019] Wakui, T., Hashiguchi, M., Sawada, K., and Yokoyama, R. (2019). Two-stage design optimization based on artificial immune system and mixed-integer linear programming for energy supply networks. *Energy*, 170:1228–1248.

[Wolsey, 1981] Wolsey, L. A. (1981). Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming*, 20(1):173–195.

[Wolsey, 2020] Wolsey, L. A. (2020). *Integer programming*. John Wiley & Sons.

[Yıldız et al., 2022] Yıldız, B., Boland, N., and Savelsbergh, M. (2022). Decomposition branching for mixed integer programming. *Operations Research*, 70(3):1854–1872.

[Zhang and Küçükyavuz, 2014] Zhang, M. and Küçükyavuz, S. (2014). Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. *SIAM Journal on Optimization*, 24(4):1933–1951.

[Zou et al., 2019] Zou, J., Ahmed, S., and Sun, X. A. (2019). Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1-2):461–502.