Decomposition of loosely coupled integer programs: A multiobjective perspective

Merve Bodur · Shabbir Ahmed · Natashia Boland · George L. Nemhauser

Abstract We consider integer programming (IP) problems consisting of (possibly a large number of) subsystems and a small number of coupling constraints that link variables from different subsystems. Such problems are called *loosely coupled* or *nearly decomposable*. In this paper, we exploit the idea of resource-directive decomposition to reformulate the problem so that it can be decomposed into a resource-directive master problem and a set of *multiobjective* programming subproblems. Recent methods developed for solving multiobjective problems enable us to formulate a relaxation of the master problem that is an IP whose solution yields a dual bound on the value of the original IP. This perspective provides a new, general framework for IP decomposition, in which many alternative algorithm designs are possible. Here, we develop one such algorithm, and demonstrate its viability and potential benefits with the aid of computational experiments knapsack-based instances with up to five coupling constraints and 7500 variables, comparing it with both a standard IP solver and a generic branch-and-price solver.

Keywords Integer programming · resource-directive decomposition · multiobjective integer programming · column generation

Merve Bodur (Corresponding author)
Department of Mechanical and Industrial Engineering, University of Toronto
E-mail: bodur@mie.utoronto.ca
Shabbir Ahmed (Deceased)
School of Industrial and Systems Engineering, Georgia Institute of Technology
Natashia Boland
School of Industrial and Systems Engineering, Georgia Institute of Technology
E-mail: natashia.boland@isye.gatech.edu
George L. Nemhauser
School of Industrial and Systems Engineering, Georgia Institute of Technology
E-mail: george.nemhauser@isye.gatech.edu

1 Introduction

We consider integer programs (IPs) of the form

$$\begin{aligned} \min\ & \sum_{i\in\mathcal{M}} (c^i)^\top x^i && && \text{(1a)}\\ \text{s.t.}\ & x^i \in \mathcal{X}^i, && i \in \mathcal{M} && \text{(1b)}\\ & \sum_{i\in\mathcal{M}} A^i x^i \le b, && && \text{(1c)} \end{aligned}$$

where the input data is defined as follows: $b \in \mathbb{R}^m$, $m \in \mathbb{Z}_+$, $\mathcal{M} := \{1, \dots, M\}$ is the index set of *blocks* and there are $M \in \mathbb{Z}_+$ blocks; for each $i \in \mathcal{M}, c^i \in \mathbb{R}^{n_i}, n_i \in \mathbb{Z}_+ \setminus \{0\}$ and $A^i \in \mathbb{R}^{m\times n_i}$; and $(\cdot)^\top$ denotes the transpose operator. We assume that $\mathcal{X}^i \subseteq \mathbb{Z}^{n_i}$, and is nonempty and bounded, for all $i \in \mathcal{M}$. For each block $i \in \mathcal{M}$, problem (1) has n_i variables, $x^i \in \mathbb{R}^{n_i}$, and constraints (1b), which include the integrality constraints. Problem (1) also has m coupling constraints, (1c), linking different blocks together. We refer to the vector b as the *resource vector*.

When $m = 0$, the model (1) is *fully decomposable* by block: it can be solved by solving M smaller IPs. We are interested in problems where the blocks are *loosely coupled*, i.e., the number of linking constraints, m, is "small".

For loosely coupled IPs, *Lagrangian relaxation* [21] may be used to obtain lower bounds on the optimal objective value. The key advantage is that when the coupling constraints are dualized, the problem decomposes by blocks. Also, stronger relaxation bounds can be obtained via Lagrangian relaxation than by linear programming (LP) relaxation of the problem. Some of the main issues with the use of Lagrangian methods are the existence of a duality gap and the difficulty in recovering primal feasible solutions. To address these issues, branch-and-bound algorithms are commonly used, where the LP relaxation is replaced by Lagrangian relaxation.

An alternative, related approach is Dantzig-Wolfe reformulation [41], which decomposes the problem into a master problem, which may have an exponential number of columns, and M subproblems. The LP relaxation of the master problem is usually solved by column generation [16], enhanced, for example, with the use of stabilization strategies, to address issues such as tailing-off and degeneracy [32]. To enforce integrality constraints in the master problem, *branch-and-price* [7] is used, with column generation carried out at each node of the branch-and-bound search tree. Branch-and-price methods are typically designed so that the pricing subproblems have special structure and only branching rules that are compatible with this structure are considered [40].

In this paper, motivated by recent developments in multiobjective programming (MOP), we develop a *MOP-based decomposition algorithm* to solve loosely coupled IPs. This approach brings a novel perspective to solving such IPs. We provide a proof of concept of this new framework and explain its relationships to existing decomposition approaches. Our computational results demonstrate the potential value of this new perspective.

We first use the idea of *resource-directive decomposition* to derive a reformulation of the problem, which allows us to establish a relationship between an optimal solution of the original IP and *nondominated points* (NDPs) in related multiobjective problems. The reformulation can be decomposed into a master problem and a set of subproblems, one *multiobjective* IP for each block. The master problem's columns correspond to NDPs of these MOP subproblems.

Multiobjective IPs may have exponentially many NDPs, but methods to solve them maintain a set of NDPs found so far and a collection of unexplored *regions* in which as yet undiscovered NDPs of the subproblem may lie. Exploiting such information, we formulate a relaxation of the resource-directive master problem, in which each column corresponds to either an NDP or an unexplored region of a MOP subproblem. This relaxation is an IP whose solution yields a lower bound on the value of the original IP.

This perspective provides a new, general framework for IP decomposition. It offers many algorithmic choices. For example, what MOP solution strategies should be used for the subproblems and how far should each be solved before the master relaxation IP is solved? How can solution of the master relaxation IP guide the MOP solution strategy to produce better lower bounds? How are the solution processes of the master and subproblems interleaved?

Here, we develop an algorithm within this framework, as a proof of concept. It generates new NDPs for the MOP subproblems by searching unexplored regions indicated by the current solution to the master relaxation IP. The set of unexplored regions is then revised using a well-known multiobjective approach. The master relaxation IP is strengthened by the addition of cutting planes derived as a byproduct of the NDP generation process and then solved to optimality.

As far as we aware, this approach to decomposition of loosely coupled IPs has not been proposed before, although it does have deep relationships to known concepts. In addition to its connections to branch and price, it has links to *value function reformulation* [1,26,37]. We explain these relationships in Section 3. In Section 5, we report on computational experiments that compare the performance of our algorithm to both a standard IP solver and a generic branch-cut-and-price solver. These discussions highlight the potential benefits of the framework and demonstrate the viability of our algorithm.

The remainder of this paper is organized as follows. Section 2 presents preliminary material on resource-directive decomposition and multiobjective optimization. Section 3 describes the proposed reformulation and how it can be decomposed, introduces the key new concepts and explains their relationships to prior related work. Section 4 describes the algorithm. Section 5 provides the numerical results, and conclusions are given in Section 6.

2 Preliminaries

Resource-directive decomposition. We first parameterize problem (1) according to how the resource vector, b, is partitioned between the blocks. Specifically, introducing *resource variables* $u^i \in \mathbb{R}^m$ for each block $i \in \mathcal{M}$ explicitly, we rewrite the problem (1) as

$$\begin{aligned}
\min \quad & \sum_{i\in\mathcal{M}} (c^i)^\top x^i \\
\text{s.t.} \quad & x^i \in \mathcal{X}^i, && i \in \mathcal{M} \\
& A^i x^i \leq u^i, && i \in \mathcal{M} \\
& \sum_{i\in\mathcal{M}} u^i \leq b, \\
& u^i \in \mathbb{R}^m, && i \in \mathcal{M}
\end{aligned}$$

which can be written equivalently as a *resource-directive master problem*

$$\text{RDMP} : \min \left\{ \sum_{i\in\mathcal{M}} f_i(u^i) : \sum_{i\in\mathcal{M}} u^i \leq b, \text{ and } u^i \in \mathbb{R}^m, i \in \mathcal{M} \right\} \tag{2}$$

where for each $i \in \mathcal{M}$, $f_i : \mathbb{R}^m \to \mathbb{R}$ is the value function of the *subproblem*:

$$\text{RDSP}(i,u): \ f_i(u) = \min \ \left\{ (c^i)^\top x : x \in \mathcal{X}^i, \ A^i x \leq u \right\}. \tag{3}$$

Note that the feasible set of a subproblem can be empty for some resource vectors, in which case the value function can be assumed to return $+\infty$.

The general idea of such a decomposition is very old. See, for example, [19,27] and references therein, notably [29], in which dynamic programming is used to find optimal values of the resource variables for linear programs.

Multiobjective optimization. We review some basic concepts in multiobjective optimization, mostly following the presentation of [17].

A multiobjective optimization problem with feasible set $\mathcal{X} \subseteq \mathbb{R}^n$ and $J \in \mathbb{Z}_+, J \geq 2$ objective functions (or criteria) $g_j : \mathcal{X} \to \mathbb{R}, j = 1, \ldots, J$ can be written as

$$\min_{x\in\mathcal{X}} \{g_1(x), \ldots, g_J(x)\}. \tag{4}$$

Define $g : \mathcal{X} \to \mathbb{R}^J$ as $g(x) = (g_1(x), \ldots, g_J(x))$; it is the *objective*, also known as the *criterion function vector*, which maps the feasible set defined in the *decision space* to the *criterion space*, $\mathbb{R}^J$. The image of the feasible set is denoted by $\mathcal{Z} := g(\mathcal{X}) := \{z \in \mathbb{R}^J : \exists x \in \mathcal{X} \text{ s.t. } z = g(x)\}$ and usually referred to as the *feasible set in criterion space*.

In multiobjective optimization, the "optimal value" is a set, often called the *nondominated* (Pareto-optimal or efficient) *frontier* (NDF). It is defined to be the set of vectors, $z \in \mathbb{R}^J$, in the criterion space, having the property that (i) z is the criterion-space image of some feasible solution, i.e., $z = g(x)$ for some $x \in \mathcal{X}$, or $z \in \mathcal{Z}$, in which case we say z is *feasible*, and (ii) there does not exist any other feasible solution, $z' \in \mathcal{Z}$, which *dominates* z, i.e., for which $z'_j \leq z_j$ for all $j = 1, \ldots, J$ and $z'_j < z_j$ for at least one index $j \in \{1, \ldots, J\}$. An element of the NDF is known as a *nondominated point* (NDP). In other words, an NDP is a feasible objective vector for which none of its components can be improved without making at least one of its other components worse. The union of preimages of NDPs is called the *efficient set*, whose elements are the *efficient solutions.* So, a feasible solution $x \in \mathcal{X}$ is *efficient* (or Pareto optimal) if its image $z = g(x)$ is an NDP, i.e., *nondominated.* On the other hand, $x \in \mathcal{X}$ is called *weakly efficient* if there is no $x' \in \mathcal{X}$ such that $g_j(x') < g_j(x)$ for all $j = 1, \dots, J$, and the point $z = g(x)$ is called *weakly nondominated.* Note that a nondominated point is a weakly nondominated point, but not vice versa.

A useful point is the so-called *ideal point*, denoted by z^I, whose components are obtained by minimizing individual objective functions over the feasible set of the problem, so $z^I_j := \min\{z_j : z \in \mathcal{Z}\}$ for all $j = 1, \dots, J$. We call the point whose components are obtained by maximizing individual objectives the *supernal point*, z^S, where $z^S_j := \max\{z_j : z \in \mathcal{Z}\}$. The NDF is contained in the hypercube defined by z^I and z^S: $z^I \leq z \leq z^S$ for all z an NDP.

An NDP can be found in a variety of ways. One of the most commonly used is the *weighted-sum method*, which solves the *weighted-sum problem* with a single objective obtained as a positive combination of the objective functions: $\min\{\lambda^\top g(x) : x \in \mathcal{X}\}$ where $\lambda_j > 0$ for all $j = 1, \dots, J$. For any positive weight vector λ, any optimal solution of this weighted-sum problem is an efficient solution for (4), i.e., its image is nondominated. Such a solution and its image are called a *supported efficient solution* and a *supported nondominated point*, respectively. Thus, an efficient solution x is supported if there exists a positive vector λ for which x is an optimal solution of the weighted-sum problem, otherwise x is called *unsupported.* We also note that if the weight vector is nonnegative, rather than positive, then an optimal solution of the scalarized problem is only guaranteed to be weakly efficient.

Another way to find an NDP is to optimize with respect to each objective function in turn, in an hierarchical manner. More specifically, we can solve *lexicographic optimization problems*, by minimizing one objective at a time, sequentially, and using optimal objective values of solved problems as constraints in the next ones. For example, for the order $1, \dots, J$, first determine $\hat{z}_1 := \min\{g_1(x) : x \in \mathcal{X}\}$, and then for each $j = 2, \dots, J$, in turn, sequentially solve $\hat{z}_j := \min\{g_j(x) : x \in \mathcal{X} \text{ and } g_{j'}(x) \leq \hat{z}_{j'}, j' = 1, \dots, j-1\}$. Then the vector $\hat{z} := (\hat{z}_1, \dots, \hat{z}_J)$ is an NDP. We represent this lexicographical optimization problem to find $\hat{z}$ as $\operatorname{lex\,min}_{x \in \mathcal{X}}(g_1(x), \dots, g_J(x))$, where the parentheses signify that the objective functions are ordered, whereas curly brackets in (4) denote that the objective functions are given as an unordered set.

Methods for solving multiobjective IPs, so as to generate the complete NDF, have developed rapidly in recent years; we refer the reader to [18] for an overview and to [8,9,10,14,23,31,42] for a representative list of recent papers.

3 Decomposition with a multiobjective perspective

Towards our reformulation of problem (1), we define the following multiobjective problems, which will be extensively used and henceforth referred to as the

MOP subproblems:

$$\begin{aligned} \mathrm{MOP}(i):\ & \min\ \{A_1^i x, \ldots, A_m^i x, (c^i)^\top x\} \\ & \text{s.t. } x \in \mathcal{X}^i, \end{aligned}$$

for $i \in \mathcal{M}$, where A_k^i denotes the k^{th} row of matrix A^i for each $k = 1, \ldots, m$. We assume that, for each block $i \in \mathcal{M}$, we can solve the single-objective IPs needed to find an NDP of MOP(i), with relative ease. We denote the feasible set of MOP(i) in criterion space by $\mathcal{Z}^i$, while we use $\mathcal{N}^i \subseteq \mathcal{Z}^i$ to represent the NDF of MOP(i), for all $i \in \mathcal{M}$. Then, we have the following observation about the resource-directive master problem, RDMP(2), whose proof is given in Appendix A.2.

Proposition 1 *Provided that RDMP has an optimal solution, there exists an optimal solution $\{\hat{u}^i\}_{i\in\mathcal{M}}$ to RDMP with the property that, for each $i \in \mathcal{M}$, the point $(\hat{u}^i, f_i(\hat{u}^i))$ is an NDP of MOP(i).*

Proposition 1 immediately implies that RDMP, (2), can be stated as:

$$\min \sum_{i\in\mathcal{M}} f_i(u^i) \tag{6a}$$

$$\text{s.t.} \sum_{i\in\mathcal{M}} u^i \le b, \tag{6b}$$

$$(u^i, f_i(u^i)) \in \mathcal{N}^i, \quad i \in \mathcal{M}, \tag{6c}$$

$$u^i \in \mathbb{R}^m, \quad i \in \mathcal{M}, \tag{6d}$$

where constraints (6c) force the point $(u^i, f_i(u^i))$ to be a nondominated point of MOP(i), for any block $i \in \mathcal{M}$. Note that formulation (6) is not equivalent to RDMP, in the sense that some feasible (even optimal) solutions of RDMP might not be feasible for (6). This is a consequence of (6c), in which $\mathcal{Z}^i$ is replaced by $\mathcal{N}^i$, the nondominated frontier. However, Proposition 1 guarantees that at least one optimal solution to RDMP is also optimal to (6).

A useful observation in the design of a decomposition algorithm is that MOP(i) does not depend on $\{u^i\}_{i\in\mathcal{M}}$. Specifically, the set $\mathcal{N}^i$, for each $i \in \mathcal{M}$, can be enumerated, at least, in principle, without the need for any particular values for the variables in problem (2), leading to the reformulation below.

For each block $i \in \mathcal{M}$, the set $\mathcal{N}^i$ is finite, as $\mathcal{X}^i$ is assumed to be a bounded set in $\mathbb{Z}^{n_i}$. We let $\mathcal{N}^i = \{z^{i,1}, z^{i,2}, \ldots, z^{i,|\mathcal{N}^i|}\}$. Also, for any vector z and a given index ℓ, we use $z_{[\ell]}$ to denote the projection of z onto the first ℓ components, i.e., $z_{[\ell]} = (z_1, \ldots, z_\ell)$. Defining binary decision variable $\lambda_k^i = 1$ if the k^{th} NDP is chosen for block $i \in \mathcal{M}$ in an optimal solution, and 0 otherwise, the model (6) can be written as

$$(\text{IP-M}) : \nu^{IP} := \min \sum_{i\in\mathcal{M}} \sum_{k=1}^{|\mathcal{N}^i|} \lambda_k^i z_{m+1}^{i,k} \tag{7a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{M}} \sum_{k=1}^{|\mathcal{N}^i|} \lambda_k^i z_{[m]}^{i,k} \leq b, \tag{7b}$$

$$\sum_{k=1}^{|\mathcal{N}^i|} \lambda_k^i = 1, \qquad i \in \mathcal{M}, \tag{7c}$$

$$\lambda^i \in \{0,1\}^{|\mathcal{N}^i|}, \qquad i \in \mathcal{M}. \tag{7d}$$

Constraints (7c) and (7d) enforce that for each block exactly one NDP of the corresponding MOP subproblem is chosen. Then, by construction of the MOP subproblems, and the definition of value functions, $f_i(\cdot)$, the objective (7a) and the resource constraints (7b) are equivalent to (6a) and (6b), respectively.

A naive exact solution method for (7) would be to first fully solve the subproblem MOP(i) to obtain the set $\mathcal{N}^i$ for each $i \in \mathcal{M}$, and then solve the model (7). An important advantage of such an approach is that it allows the MOP subproblems to be solved asynchronously. Also, any MOP algorithm can be used to generate the NDFs. Another advantage, is that efficient solutions, i.e., feasible solutions in terms of the x variables, of the MOP subproblems need not be revealed. In other words, MOP algorithms can be used as black boxes to provide the objective function (including resource usage) values. Keeping the solutions private is an important role in some applications.

The biggest challenge in solving (IP-M) is the potentially large size of the NDFs. With pure integer data and a fixed number of objectives, the number of NDPs is at worst pseudopolynomial in the size of the MOP. However, in general, it can be exponentially large, and is certainly, in the worst case, exponential in m, which may prevent the solution of (IP-M) by a standard IP algorithm. However, most of the variables in (IP-M) will take value zero in an optimal solution. This suggests a possibility of dynamically generating only those NDPs "needed". Before introducing ideas for how this may be done, we first discuss the relationships between this and previous approaches.

3.1 Relationship to Dantzig-Wolfe decomposition

The traditional Dantzig-Wolfe decomposition ([15]) of a problem having the form of (1) yields an LP, called the *master* problem, that can be written as

$$\text{DW-LP} : \min \quad \sum_{i \in \mathcal{M}} \sum_{k=1}^{|\mathcal{E}^i|} \lambda_k^i (c^i)^\top s^{i,k} \tag{8a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{M}} \sum_{k=1}^{|\mathcal{E}^i|} \lambda_k^i A^i s^{i,k} \leq b, \tag{8b}$$

$$\sum_{k=1}^{|\mathcal{E}^i|} \lambda_k^i = 1, \qquad i \in \mathcal{M}, \tag{8c}$$

$$\lambda^i \geq \mathbf{0}, \qquad i \in \mathcal{M}, \tag{8d}$$

where $\mathcal{E}^i$ is the set of extreme points of $\text{conv}(\mathcal{X}^i)$ and $s^{i,k} \in \mathcal{E}^i$ denotes the kth such extreme point. To address the potentially large number of variables, this LP is usually solved by column generation, which dynamically generates elements of $\mathcal{E}^i$, by solving a *pricing problem*: given the optimal LP dual variables $(\hat{\gamma}, \hat{\alpha})$, corresponding to (8b) and (8c) respectively, for a *restricted master* LP over only a *subset* of $\mathcal{E}^i$, the subproblem IP

$$\tilde{r}_i := \min_{s \in \mathcal{E}^i} \left\{ (c^i)^\top s - \hat{\gamma}^\top A^i s + \hat{\alpha}_i \right\} = \hat{\alpha}_i + \min_{x \in \mathcal{X}^i} \left\{ \left((c^i)^\top - \sum_{j=1}^m \hat{\gamma}_j A^i_j \right) x \right\} \tag{9}$$

is solved for each $i \in \mathcal{M}$ to determine the most negative reduced cost extreme point. For at least one $i \in \mathcal{M}$ with $\tilde{r}_i < 0$, an optimal solution of the pricing subproblem for i is added to the restricted master LP. This LP is re-solved, and the process iterated, until $\tilde{r}_i \geq 0$ for all $i \in \mathcal{M}$.

Here $x^i := \sum_{k=1}^{|\mathcal{E}^i|} \lambda^i_k s^{i,k}$ for each $i \in \mathcal{M}$ recovers the original variables of the IP, which are required to be integer since $\mathcal{X}^i \subseteq \mathbb{Z}^{n_i}$. In general, the solution to the master LP does not satisfy this requirement, and branching is required. Branching is usually carried out in the space of the original variables, with branching constraints applied to individual x variables so as to not disturb special structure in the pricing subproblem (special structure in $\mathcal{X}^i$). This type of branching and the reasons for it were discussed as early as the first papers on branch-and-price ([7,34]); see also [16]. After such branching, the sets $\mathcal{X}^i$ change, and so may the sets $\mathcal{E}^i$: a set $\mathcal{E}^i$ for a problem deeper in the branch-and-price tree is not necessarily a subset of the corresponding set at the root node. Example 1 in Appendix C.1 exhibits this phenomenon. Importantly, this phenomenon is not an artifact of the restricted master problem, but may occur even when the master LP includes every extreme point and so is not a restriction.

Expressing $\xi^{i,k} := \begin{bmatrix} A^i \\ (c^i)^\top \end{bmatrix} s^{i,k}$ and substituting $\xi^{i,k}_{m+1}$ for $(c^i)^\top s^{i,k}$ in the objective (8a) and $\xi^{i,k}_{[m]}$ for $A^i s^{i,k}$ in the resource constraint (8b) for each i and k exposes the similarity in the structure of MOP-based reformulation (IP-M) and the Dantzig-Wolfe reformulation. Their difference hinges on

1. the difference between their integrality constraints, and
2. the difference between $\mathcal{N}^i$ and $\Phi^i := \left\{ \begin{bmatrix} A^i \\ (c^i)^\top \end{bmatrix} s^{i,k} : s^{i,k} \in \mathcal{E}^i \right\}$, which is the set of images of $\mathcal{E}^i$ in the objective space of MOP(i), representing the set of distinct columns that may appear in (DW-LP).

In the case that $\mathcal{X}^i$ is a pure binary set, every element of $\mathcal{X}^i$ is an extreme point of $\text{conv}(\mathcal{X}^i)$, so $\mathcal{E}^i = \mathcal{X}^i$. As a consequence, the Dantzig-Wolfe integrality requirement can be modeled as $\lambda^i \in \{0,1\}^{|\mathcal{E}^i|}$, having the same form as (7d). Thus, in this case, their difference lies primarily in the relationship between $\mathcal{N}^i$ and Φ^i. For $\mathcal{X}^i$ pure binary, $\mathcal{N}^i \subseteq \Phi^i$, and it can be expected that, in general, the MOP-based reformulation (IP-M) will contain fewer columns than the Dantzig-Wolfe reformulation. The fact that the difference may be substantial, with $|\mathcal{N}^i| << |\Phi^i|$, is illustrated in Appendix C.1, Example 2, which provides a simple family of examples with $|\Phi^i| - |\mathcal{N}^i| = 3M$, so the difference grows linearly in M, the number of blocks.

Interestingly, the column generation process may prevent generation of many points in $\Phi^i \setminus \mathcal{N}^i$, even when this set is very large. Dual feasibility for the restricted master LP ensures $\hat{\gamma} \leq \mathbf{0}$, hence the objective function in the column generation pricing subproblem, (9), can be expressed as

$$(\beta)^\top \begin{bmatrix} A^i \\ (c^i)^\top \end{bmatrix} x \quad \text{where} \quad \beta := \begin{bmatrix} -\hat{\gamma} \\ 1 \end{bmatrix} \geq \mathbf{0}. \tag{10}$$

Thus the pricing subproblem is a nonnegative scalarization of the multiple objectives for MOP(i), and so – at least at the root node of the branch-and-price process – must yield a weakly nondominated point, which is not necessarily nondominated. Specifically, whenever $\hat{\gamma}$ has no zero elements, so every element of the vector is strictly negative, the column generation pricing problem at the root node ensures that only points in $\mathcal{N}^i$ can be generated. (In fact, only *supported* NDPs can be generated by the pricing problem at the root node; it is impossible to generate unsupported NDPs with a nonnegative scalarization.) Otherwise, if $\hat{\gamma}$ has some zero element, the pricing problem may return a solution with image not in $\mathcal{N}^i$. This situation is illustrated in Example 3, Appendix C.1. At nodes other than the branch-and-price root node, even columns that are not weakly nondominated can be generated. Indeed, traditional branching in branch-and-price can lead to exploration of nodes for which no column in $\mathcal{N}^i$ is feasible for the pricing subproblem, for some i. A case in which this occurs is discussed in Example 4.

In the more general setting, with integer variables, some of which are not necessarily binary, the difference between the Dantzig-Wolfe and MOP-based reformulations may also be in the integrality constraint. Integrality for (IP-M) is given explicitly as (7d); the original IP can be solved by solving (IP-M). In general, this not true for the Dantzig-Wolfe reformulation. The integrality requirement that $\sum_{k=1}^{|\mathcal{E}^i|} \lambda_k^i s^{i,k} \in \mathbb{Z}^{n_i}$ for all $i \in \mathcal{M}$ requires branching to "expose" solutions that are not extreme points of conv$(\mathcal{X}^i)$ at the outset. This is observed in [41], where extreme points are replaced by the notion of *generating sets* to address the issue. We give an illustration of it in Example 1, Appendix C.1. As a consequence of this issue, the difference between Dantzig-Wolfe and MOP-based decomposition is quite profound: in the general case, Dantzig-Wolfe decomposition does *not* provide a reformulation of the IP (1), whereas MOP-based decomposition does.

3.2 Relationship to value function reformulation

In the context of two-stage stochastic integer programming, decomposable structure has inspired a reformulation related to the MOP-based (IP-M) reformulation, (7). Originating with the work of [1] and [26], the problem structure is of the form

$$
\begin{aligned}
\text{(2SSIP)}: \quad & \min \sum_{i\in\mathcal{M}} (c^i)^\top x^i && && \text{(11a)}\\
& \text{s.t. } x^i \in \mathcal{X}^i, && i \in \mathcal{M} && \text{(11b)}\\
& \qquad Tx^1 + D^i x^i \le h^i, && i = 2,\dots,M, && \text{(11c)}
\end{aligned}
$$

where variables x^1 represent the first-stage variables and x^i the second-stage variables under the $(i-1)$th scenario. The special form of the linking constraints, (11c), permits the resource-directive master problem of (2SSIP) to be greatly simplified, to

$$
\min \left\{ \sum_{i\in\mathcal{M}} f_i(u^i) \ : \ u^1 + u^i \le h^i, \ i = 2,\dots,M \right\}, \tag{12}
$$

which is referred to as the *value function reformulation*. Here $f_i(u)$ is the value function defined as in (3), with $A^1 := T$ and $A^i := D^i$ for $i = 2,\dots,M$. In [1], (12) is solved by a branch-and-bound method that searches over the space of u^1 variables, partitioning this space into hyper-rectangles. The corners of the hyper-rectangle are used to calculate a lower bound: for $u^1 \in [\underline{u}^1, \overline{u}^1]$, a lower bound is given by $f_1(\overline{u}^1) + \sum_{i=2}^{M} f_i(h^i - \underline{u}^1)$. The branching scheme splits the hyper-rectangle for u^1 in two, along one axis, so as to cut off this lower bound.

In [37], it is recognized that only nondominated points of MOP(i) for each $i \in \mathcal{M}$ need to be considered. This follows from the equivalence of the *level-set minimal vectors* introduced in [37] and NDPs; we give proof of this in Appendix A.3. Thus [37] obtain the reformulation[1] for (2SSIP):

$$
\min \left\{ \sum_{i\in\mathcal{M}} f_i(u^i) \ : \ u^1 + u^i \le h^i, \ i = 2,\dots,M, \ (u^i, f_i(u^i)) \in \mathcal{N}^i, \ i \in \mathcal{M} \right\}.
$$

However, instead of working with NDPs, [37] employs a relaxation of $\mathcal{N}^i$ called the *integral monoid*, which is equivalent to the image in MOP(i) objective space of $\mathcal{X}^i$. A branch-and-bound search in the space of u^1 based on partitioning hyper-rectangles in two is also used in [37], with several enhancements. As well as improved lower bounding, the method of [37] cleverly leverages properties of value functions in preprocessing and to reduce computational burden in value function calculation, with very substantial impact.

In what follows, we, too, propose to search over the resource variables, u^i, and make use of hyper-rectangles, but with some key differences:

1. due to the very different form of the resource constraints ($u^1 + u^i \le h^i$ for $i = 2,\dots,M$ versus $u^1 + u^2 + \cdots + u^M \le b$), we must search over u^i for *all* $i \in \mathcal{M}$, not just u^1, and

[1] We have slightly generalized the ideas of [37], which in their original form only needed two sets, $\mathcal{N}^1$ and $\mathcal{N}^2$, since they considered the case of stochasticity in the right-hand side only, so their recourse matrices D^i for $i = 2,\dots,M$ are identical.

2. our partitioning of the space is designed to ensure that only NDPs are considered.

3.3 Towards an algorithm: discussion

We propose to solve the RDMP (2) without enumerating all NDPs of the MOP subproblems. Clearly this is a challenge. It would be natural to attempt branch-and-price, and, indeed, as we explain in detail later, solving the root node LP relaxation of (7) can be done with column generation, after a minor modification to the pricing problem. However, branching is a serious obstacle: branching on the master problem variables faces the same issues as in standard branch-and-price (see e.g., [34]), while branching on the original variables can cut off NDPs (shown in Example 4, Appendix C.2).

One branching scheme that would preserve NDPs exploits the NDP definition: if z^i is an NDP of MOP(i), and so is y, then the following disjunction is satisfied:

$$(y = z^i) \ \vee \ (y_1 < z^i_1) \ \vee \ \ldots \ \vee (y_m < z^i_m) \ \vee (y_{m+1} < z^i_{m+1}).$$

Thus if a solution to the LP relaxation of (IP-M), $\tilde{\lambda}$, say, has $\tilde{\lambda}^i$ fractional for some i, we may select j to be any index for which there are least two distinct values in the set $\{z^{i,k}_j : \tilde{\lambda}^i_k > 0\}$, and then choose k^* so that the NDP z^{i,k^*}_j minimizes $z^{i,k}_j$ over all k with $\tilde{\lambda}^i_k$ nonzero. (Such a j must exist, since otherwise the support of $\tilde{\lambda}^i$ must be a singleton, contradicting fractionality of $\tilde{\lambda}^i$.) Then the disjunction

$$(y_1 < z^{i,k^*}_1) \vee \ldots \ (y_{j-1} < z^{i,k^*}_{j-1}) \vee (y_j \leq z^{i,k^*}_j) \vee \ (y_{j+1} < z^{i,k^*}_{j+1}) \vee \ldots \vee (y_{m+1} < z^{i,k^*}_{m+1})$$

is satisfied by any y an NDP of MOP(i). Furthermore, for each term in the disjunction, there is at least one NDP $z^{i,k}$ with $\tilde{\lambda}^i_k > 0$ which does not satisfy this term: z^{i,k^*} satisfies only the term for j, and there must exist another NDP, $z^{i,k}$, say with $z^{i,k}_j$ distinct from z^{i,k^*}_j, and hence with $z^{i,k}_j > z^{i,k^*}_j$. Thus $z^{i,k}$ cannot satisfy the jth term. Consequently, if this disjunction is used for branching, the current LP solution cannot recur.

We call this branching scheme *NDP-based branching*. It is an appealing branching scheme in one respect: it preserves the MOP subproblem structure. This follows from the following lemma, which is well known in multiobjective optimization. For completeness, we provide a brief proof in Appendix A.1.

Lemma 1 *For a given $\delta \in \mathbb{R}^J$, if $z \in \mathbb{R}^J$ is an NDP of*

$$\min \ \{g_1(x), \ldots, g_J(x)\} \tag{13a}$$

$$s.t. \ x \in \mathcal{X} \tag{13b}$$

$$g_j(x) \leq \delta_j, \quad j \in \mathcal{J}' \tag{13c}$$

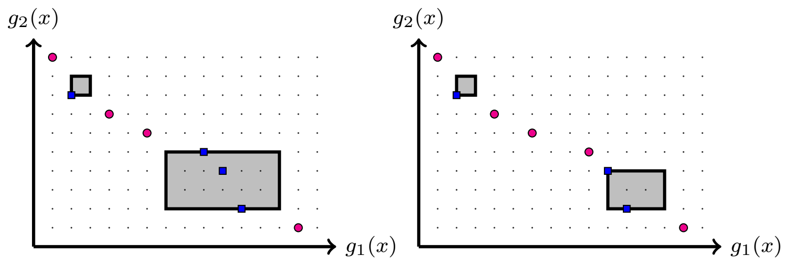
where $\mathcal{J}' \subseteq \{1, \ldots, J\}$, then z is also an NDP of (4).

Thus adding branching constraints of the form $z^i_j < z^{i,k^*}_j$ to MOP(i), which (as is typical in multiobjective optimization) are implemented as $z^i_j \leq z^{i,k^*}_j - \epsilon$ for a suitable tolerance $\epsilon > 0$, preserves NDPs.

This branching scheme has the potential to produce far smaller branch-and-bound trees than traditional branch-and-price, which branches on the original variables, x^i_j for some $i \in \mathcal{M}$, $j \in \{1, \dots, n_i\}$. Integer programming folklore suggests that branching first on higher level information – "big picture" decisions – leads to smaller search trees. NDP-based branching branches in the space of the u variables of the RDMP, which carries higher level information than branching in the space of the original variables, as in traditional branch-and-price. In the presence of symmetry or near-symmetry leading to alternate optima – multiple solutions in the space of the original variables mapping to the same NDP or to weakly dominated points in the MOP objective space – branching in the MOP objective space can be particularly effective. This is well illustrated in Example 4 (Appendix C.2): an instance with two blocks subproblems, each an assignment problem, and a single linking constraint requires in excess of 103 nodes to solve with traditional branch-and-price branching, while NDP-based branching requires only 11 nodes. The presence of alternate optima in this example can be seen in the chains of branches with child nodes having the same LP objective value as their parent in the traditional branch-and-price search tree.

However, NDP-based branching has two drawbacks. One is that it only covers, and does not partition, the set of NDPs. The other, more serious drawback, is that it introduces multi-way branching, with $m + 1$ child nodes from each parent node in the branch-and-bound tree. The use of multi-way branching in branch-and-bound is still largely unexplored, a gap in the field highlighted in the survey article [28].

To avoid the necessity of designing an effective multi-way branch-and-bound procedure, we propose an alternative approach to solving RDMP, which embeds multi-way disjunctions within an IP model. The key to unlocking this approach is the observation that any subset, $\mathcal{P}$, of a NDF of a MOP induces a *disjunctive relaxation* of the NDF: the NDPs in $\mathcal{P}$ can be used to construct a set of polyhedral regions, $\{\mathcal{Q}_1, \dots, \mathcal{Q}_q\}$, in the criterion space that the remaining undiscovered NDPs belong to. In other words, given a set of NDPs found so far, $\mathcal{P}$, the NDF lies within $\Omega := \mathcal{P} \cup \bigcup_{r=1}^{q} \mathcal{Q}_r$. This relaxation is well known and extensively used in criterion space methods for multiobjective IP [9,14]. Closed forms for the linear constraints defining the polyhedral regions $\{\mathcal{Q}_1, \dots, \mathcal{Q}_q\}$ are deduced by a process equivalent to iterative application of the NDP-based branching rule for each NDP in $\mathcal{P}$ [33,14]. In Figure 1, we illustrate two disjunctive relaxations of the NDF of an integer program with two objectives ($g_1(x)$ and $g_2(x)$), obtained from two different sets $\mathcal{P}$, where circle points, square points, shaded regions and dots correspond to discovered NDPs, undiscovered NDPs, regions and integer points, respectively, with tolerance $\epsilon = 1$, which is suitable in the case of pure integer data.



(a) $|\mathcal{P}| = 4, q = 2$ (b) $|\mathcal{P}| = 5, q = 2$

Fig. 1 Decomposition of the criterion search space of a biobjective integer program into a) five NDPs and two rectangular regions b) four NDPs and two rectangular regions

We thus propose an algorithm in which, for each subproblem MOP(i), we have a set of NDPs discovered so far, denoted by $\mathcal{P}^i \subseteq \mathcal{N}^i$, inducing a disjunctive relaxation of the NDF of MOP(i), given by a set of polyhedra, $\Lambda^i = \{\mathcal{Q}^i_1, \ldots, \mathcal{Q}^i_{q_i}\}$, in $\mathbb{R}^{m+1}$, with the property that

$$\mathcal{N}^i \subseteq \Omega^i := \mathcal{P}^i \cup \bigcup_{r=1}^{q_i} \mathcal{Q}^i_r \,. \tag{14}$$

Then the following *disjunctive programming* problem gives a lower bound on (IP-M):

$$\text{(LB-DJ)}: \nu^{LB-DJ} := \min \sum_{i \in \mathcal{M}} w^i \tag{15a}$$

$$\text{s.t.} \sum_{i \in \mathcal{M}} u^i \leq b \tag{15b}$$

$$(u^i, w^i) \in \mathcal{P}^i \vee \bigvee_{r=1,\ldots,q_i} (u^i, w^i) \in \mathcal{Q}^i_r, \quad i \in \mathcal{M}. \tag{15c}$$

Note that since $\mathcal{P}^i$ is a discrete set, the disjunctive constraint (15c) in fact has $|\mathcal{P}^i| + q_i$ disjunctive terms for each $i \in \mathcal{M}$.

There have been substantial advances in disjunctive programming over recent decades. Since the pioneering work of E. Balas in theory and algorithms (e.g., [2,3,5,6]), with a very thorough treatise published recently [4], concepts and general paradigms for disjunctive programming have been making major impacts in practice, not least through the work of I. Grossmann (e.g. [13,22, 38,39]). In the algorithm we have implemented, we model (LB-DJ) as a MIP of the stronger, disaggregated, type described as "(CH)" in [35] (details in Section 4.1). However, other approaches to solving (LB-DJ) may readily be substituted in the algorithmic framework we propose, which will benefit as further advances in disjunctive programming arise, as, for example, in [24].

3.4 Algorithm overview

Our algorithm alternates between solving (LB-DJ) to obtain a lower bound on (IP-M) and refining the polyhedral regions, $\mathcal{Q}^i$, until a solution $(\hat{u}^i, \hat{w}^i)_{i\in\mathcal{M}}$ corresponding to the lower bound uses only NDPs, i.e., has $(\hat{u}^i, \hat{w}^i) \in \mathcal{P}^i$ for all $i \in \mathcal{M}$; this must be an optimal solution to (IP-M). Otherwise, if $(\hat{u}^i, \hat{w}^i) \in \mathcal{Q}^i_r$ for some i and r, then an NDP in $\mathcal{Q}^i_r$ is found by minimizing a (positive) weighted sum of the MOP(i) objectives over $\mathcal{Q}^i_r$ (or it is proved that no such NDP exists, in which case $\mathcal{Q}^i_r$ can be removed and the relaxation tightened). This minimization provides a cut that can be used to tighten $\mathcal{Q}^i_r$. Whether or not this cut cuts off $(\hat{u}^i, \hat{w}^i)$, the new NDP found can be added to $\mathcal{P}^i$ and a new, refined, disjunctive relaxation of MOP(i) is induced, ensuring that the updated (LB-DJ) is a tighter disjunctive programming relaxation of (IP-M). Cuts of the type we describe have been used in recent algorithms for multiobjective integer programming, e.g. [36], to tighten regions of the criterion space in which the NDF is not yet known. In the case that the weights in the weighted-sum minimization correspond to dual variables for the resource constraints in a MIP formulation of (LB-DJ), the cut resulting from this minimization corresponds to the Lagrange cut [11,12][2].

A flow chart of our algorithm, which we call the *MOP-based decomposition algorithm*, is provided in Figure 2. The box with dashed borders, "Seek UB", represents an optional step in which an upper bound is sought. For example, the disjunctive program formed by replacing (15c) in (LB-DJ) by the requirement that $(u^i, w^i) \in \mathcal{P}^i$ for all $i \in \mathcal{M}$ could be solved (perhaps heuristically). If that process is always skipped, then the algorithm only finds a feasible solution when it finds an optimal solution.

The algorithm described so far admits of many incarnations and natural generalizations. For example, it is not necessary that (LB-DJ) be solved to optimality; all that is required is that (u, w) returned from the solution process for (LB-DJ) corresponds to a lower bound on the value of (LB-DJ). The best way to engineer a specific algorithm from this broad class is left to future research. In the next section, we discuss a specific incarnation of it.

4 A MOP-based decomposition algorithm

In this section, we introduce upper and lower bounding problems, explain details of the algorithm, prove finite convergence and then suggest some algorithmic enhancements and options.

[2] This remark needs to be interpreted somewhat carefully. The correspondence only applies to a column generation process that is *local* to the polyhedral region.

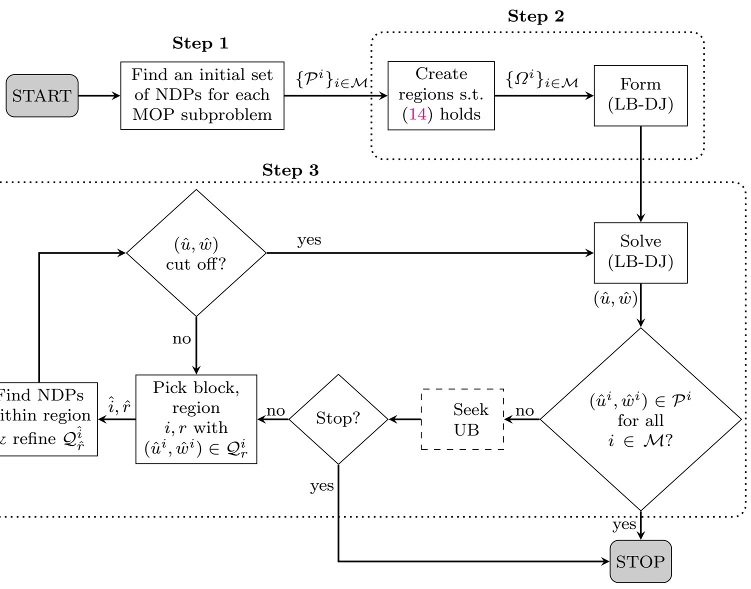

Step 1
Step 2
Step 3
START
Find an initial set of NDPs for each MOP subproblem
$\{\mathcal{P}^i\}_{i\in\mathcal{M}}$
Create regions s.t. (14) holds
$\{\Omega^i\}_{i\in\mathcal{M}}$
Form (LB-DJ)
Solve (LB-DJ)
$(\hat{u},\hat{w})$
$(\hat{u},\hat{w})$ cut off?
yes
no
$(\hat{u}^i,\hat{w}^i)\in\mathcal{P}^i$ for all $i\in\mathcal{M}$?
no
Seek UB
Stop?
no
yes
Pick block, region i,r with $(\hat{u}^i,\hat{w}^i)\in\mathcal{Q}^i_r$
$\hat{i},\hat{r}$
Find NDPs within region & refine $\mathcal{Q}^{\hat{i}}_{\hat{r}}$
yes
STOP

Fig. 2 Flow chart of the MOP-based decomposition algorithm. $(\hat{u},\hat{w})$ denotes an optimal solution of the (LB-DJ) problem.

4.1 Upper and lower bounding problems

Suppose that each MOP(i) has been (partially) solved to obtain a set of NDPs, $\mathcal{P}^i \subseteq \mathcal{N}^i$, and a set of polyhedra, $\Lambda^i = \{\mathcal{Q}^i_1,\dots,\mathcal{Q}^i_{q_i}\}$, in $\mathbb{R}^{m+1}$, with the property that (14) holds. Without loss of generality, we assume that $\mathcal{N}^i = \{z^{i,1}, z^{i,2},\dots,z^{i,|\mathcal{N}^i|}\}$ is ordered such that its first $|\mathcal{P}^i|$ elements are the ones of $\mathcal{P}^i$, for all $i\in\mathcal{M}$. Then, the integer program obtained by restricting $\mathcal{N}^i$ in (IP-M) to its first $|\mathcal{P}^i|$ elements, for each $i\in\mathcal{M}$, given by

$$(\text{UB-M}) : \nu^{UB} := \min \sum_{i\in\mathcal{M}}\sum_{k=1}^{|\mathcal{P}^i|} \lambda^i_k z^{i,k}_{m+1} \tag{16a}$$

$$\text{s.t.} \sum_{i\in\mathcal{M}}\sum_{k=1}^{|\mathcal{P}^i|} \lambda^i_k z^{i,k}_{[m]} \le b \tag{16b}$$

$$\sum_{k=1}^{|\mathcal{P}^i|} \lambda^i_k = 1, \qquad i\in\mathcal{M} \tag{16c}$$

$$\lambda^i \in \{0,1\}^{|\mathcal{P}^i|}, \qquad i\in\mathcal{M} \tag{16d}$$

gives an upper bound on the optimal value ν^{IP} of (IP-M), which becomes exact when $\mathcal{P}^i = \mathcal{N}^i$ for all $i\in\mathcal{M}$. Note that (UB-M) can be infeasible, in which case the upper bound is taken to be $+\infty$. We also note that if all data in (UB-M) is nonnegative and integer, then $|\mathcal{P}^i|$ is $O(\prod_{j=1}^m b_j)$, equivalently

$O(B^m)$ where $B := \max_{j=1,\dots,m} b_j$. In this case, it is easy to see that (UB-M) can be solved in $O(MB^m)$ time by dynamic programming, which implies that (UB-M) can be solved in pseudopolynomial time in m.

Using relaxations $\Omega^i, i \in \mathcal{M}$, defined in (14), and modeling the disjunctive program (LB-DJ) as a mixed integer program in the manner discussed in [35], for example, we obtain the lower bounding problem:

$$
\begin{aligned}
\text{(LB-M)}: \nu^{LB} := \min\ & \sum_{i\in\mathcal{M}} \left(\sum_{k=1}^{|\mathcal{P}^i|} \lambda_k^i z_{m+1}^{i,k} + \sum_{r=1}^{q_i} w_r^i \right) && \text{(17a)}\\
\text{s.t.}\ & \sum_{i\in\mathcal{M}} \left(\sum_{k=1}^{|\mathcal{P}^i|} \lambda_k^i z_{[m]}^{i,k} + \sum_{r=1}^{q_i} u_r^i \right) \le b && \text{(17b)}\\
& \sum_{k=1}^{|\mathcal{P}^i|} \lambda_k^i + \sum_{r=1}^{q_i} \mu_r^i = 1, \qquad i \in \mathcal{M} && \text{(17c)}\\
& \mu_r^i = 1 \Rightarrow (u_r^i, w_r^i) \in \mathcal{Q}_r^i, \qquad i \in \mathcal{M},\ r = 1,\dots,q_i && \text{(17d)}\\
& \mu_r^i = 0 \Rightarrow (u_r^i, w_r^i) = (\mathbf{0}, 0), \quad i \in \mathcal{M},\ r = 1,\dots,q_i && \text{(17e)}\\
& \lambda^i \in \{0,1\}^{|\mathcal{P}^i|}, \mu^i \in \{0,1\}^{q_i} \quad i \in \mathcal{M}, && \text{(17f)}\\
& u_r^i \in \mathbb{R}^m,\ w_r^i \in \mathbb{R}, \qquad i \in \mathcal{M},\ r = 1,\dots,q_i. && \text{(17g)}
\end{aligned}
$$

For block $i \in \mathcal{M}$, binary variable λ_k^i represents the selection of the k^{th} NDP as before, while binary variable μ_r^i denotes the selection of r^{th} region, $\mathcal{Q}_r^i$. Continuous variables u_r^i and w_r^i correspond to the resource and cost components of a point chosen from the region $\mathcal{Q}_r^i$, respectively. Constraints (17c) state that, for each block, either a previously generated NDP or a region is chosen in the solution (together with the domain constraints (17f) enforce SOS1 restrictions). In the latter case, the logical constraints (17d) ensure that the chosen point belongs to the region. Otherwise, due to constraints (17e), no point from the region appears in the solution. Depending on the point selection for a block, the terms in parentheses in (17a) and (17b) signify the corresponding cost and the resource consumption, respectively.

The logical constraints (17d) and (17e) can be represented by a set of linear inequalities, as the regions are assumed to be polyhedral and can be bounded, since the MOP feasible sets are assumed bounded. Specifically, since $\mathcal{X}^i$ is assumed bounded, for each $i \in \mathcal{M}$ and each region $r = 1,\dots,q_i$, there are vectors $\underline{\xi}_r^i, \overline{\xi}_r^i \in \mathbb{R}^{m+1}$ so that $\underline{\xi}_r^i \le z \le \overline{\xi}_r^i$ for any $z \in \mathcal{Q}_r^i \cap \mathcal{N}^i$. For example, $\underline{\xi}_r^i$ may be taken to be the ideal point of MOP(i) and $\overline{\xi}_r^i$ taken to be its supernal point, for all r. Then, if each region is given as a polyhedron, $\mathcal{Q}_r^i = \{z \in \mathbb{R}^{m+1} : D_r^i z \ge d_r^i\}$, where D_r^i and d_r^i are a matrix and vector of appropriate dimension, for each i and r, the logical constraints (17d) and (17e) can be modeled linearly, (as in the “(CH)” model of [35]), as

$$D_r^i(u_r^i, w_r^i) \ge d_r^i \mu_r^i, \qquad i \in \mathcal{M},\ r = 1,\dots,q_i \tag{18a}$$

$$\underline{\xi}_r^i \mu_r^i \leq (u_r^i, w_r^i) \leq \overline{\xi}_r^i \mu_r^i, \qquad i \in \mathcal{M},\ r = 1, \ldots, q_i. \tag{18b}$$

The inclusion of lower bounds on the w_r^i variables ensures that (LB-M) is bounded. Lastly, we note that if each region $\mathcal{Q}_r^i$ is represented by its ideal point, then (LB-M) can also be solved in pseudopolynomial time by dynamic programming, similar to (UB-M).

In the next proposition, whose proof is provided in Appendix A.4, we show that (LB-M) is indeed a valid lower bounding problem.

Proposition 2 $\nu^{LB} \leq \nu^{IP}$.

4.2 Details and enhancements of the MOP-based decomposition algorithm

In this section, we provide details of our MOP-based decomposition algorithm, and discuss some possible enhancements. The main structure of the algorithm is given in Algorithm 1. We next elaborate on each of its three phases. However, we first observe that in Phase 1 and in step (iii)(b) of Phase 3, Lemma 1 is used: if needed, we modify the MOP subproblems by including some upper bound constraints of type (13c). In other words, we also assume that the following constraints are included in MOP(i):

$$(c^i)^\top x \leq \delta_{m+1}^i \text{ and } A_j^i x \leq \delta_j^i, \quad j = 1, \ldots, m,$$

where $\delta^i \in (\mathbb{R} \cup \{+\infty\})^{m+1}$ is a given upper bound vector. Unless specifically mentioned, we have $\delta^i = \{+\infty\}^{m+1}$.

Algorithm 1 MOP-based decomposition. ϵ_{gap} denotes the given threshold for the absolute optimality gap.

```
1. Solve the LP relaxation of (IP-M) via (a modified form of) column generation
2. Using the NDPs generated in the previous step, construct a disjunctive relaxation of
   the NDF of each MOP($i$), creating regions so that (14) is satisfied, and form (LB-M).
3. do
        (i) Solve (LB-M)
       (ii) Solve (UB-M)   [optional, e.g., at every $\tau$ iterations]
      (iii) do
                a) Select a block for which (LB-M) has selected a region
                b) Refine this region, after generating new NDPs and/or valid inequalities,
                   ensuring that (14) remains valid
            until the current optimal solution of (LB-M) has been cut off
       (iv) Refine all regions that include any of the newly generated NDPs   [optional]
   until $\nu^{UB} - \nu^{LB} \leq \epsilon_{gap}$ or another stopping criterion has been reached
```

4.2.1 Computing initial NDPs by column generation (Phase 1)

In order to create an initial set of NDPs, we solve the LP relaxation of (IP-M). This can be done by a modified form of column generation, which we call *Pareto column generation.* We start with a set of columns – an initial set $\mathcal{P}^i$ for each $i \in \mathcal{M}$ – that is guaranteed to yield a feasible LP solution. (In our implementation, we use dummy columns: we initialize the LP with one column for each MOP, consisting of its ideal point, but with a high objective coefficient in (IP-M). Provided the LP relaxation of (IP-M) is feasible, the dummy columns will not be used in the LP solution at the completion of the column generation procedure and can be dropped from the sets $\mathcal{P}^i$; otherwise the LP relaxation of (IP-M) must be infeasible and hence so must the original IP.)

At every Pareto column generation iteration, we solve the LP relaxation of (IP-M), *restricted* to the set of columns in $\mathcal{P}^i$, $i \in \mathcal{M}$, to get $\hat{\gamma}$ and $\hat{\alpha}$, the optimal dual multipliers of the resource linking constraints (16b) and the convexity constraints (16c), respectively. Then, for each block $i \in \mathcal{M}$, we solve the *pricing problem*

$$\tilde{r}_i := \min_{z \in \mathcal{Z}^i} \{z_{m+1} - \hat{\gamma}^\top z_{[m]} - \hat{\alpha}_i\} \quad \equiv \quad \min_{x \in \mathcal{X}^i} \{(c^i)^\top x + \sum_{j=1}^{m} \hat{\gamma}_j A^i_j x - \hat{\alpha}_i\} \tag{19}$$

in order to find a column with the most negative reduced cost. If $\tilde{r}_i \geq 0$ then no column for block i has negative reduced cost, and no change is made to $\mathcal{P}^i$. Otherwise, when $\tilde{r}_i < 0$, we add some $z \in \mathcal{Z}^i$ that is optimal for the pricing problem to $\mathcal{P}^i$. So far, this is standard column generation: the modification concerns the choice of pricing problem solution to add.

If $\hat{\gamma} < \mathbf{0}$, meaning that $\hat{\gamma}_j < 0$ for all $j = 1, \ldots, m$, then *any* $z \in \mathcal{Z}^i$ that is optimal for the pricing problem is a *supported* NDP of MOP(i) and may be added to $\mathcal{P}^i$. Otherwise, (if $\hat{\gamma}_j = 0$ for some j), some optimal solutions to the pricing problem may not be NDPs: they are only guaranteed to be weakly nondominated. In such a case, a second subproblem optimization is required to find an optimal solution that is an NDP. In our implementation, letting $\tilde{z}^i \in \mathcal{Z}^i$ denote an arbitrary optimal solution to the pricing problem, we solve the following integer program, where $\mathbf{1}$ is the vector of all ones,

$$\min_z \{\mathbf{1}^\top z : z \in \mathcal{Z}^i \text{ and } z \leq \tilde{z}^i\} \tag{20}$$

to get an optimal solution, denoted by $\bar{z}^i$. This is guaranteed to be an NDP of MOP(i), by Lemma 1 and since all objective coefficients in (20) are positive. Since $\hat{\gamma} \leq \mathbf{0}$ and $\bar{z}^i \leq \tilde{z}^i$, it must be that $\bar{z}^i$ is also an optimal solution to the pricing problem (19) and so has most negative reduced cost; it is added to $\mathcal{P}^i$. Note that the MIPs (19) and (20) are assumed to be tractable, as they correspond to a weighted-sum version of MOP(i) with $\delta^i = \{+\infty\}^{m+1}$ and $\delta^i = \tilde{z}^i$, respectively.

After the addition of new columns, the LP relaxation of the restricted (IP-M) is re-solved, and the process repeated. The column generation algorithm stops when all pricing problems have nonnegative optimal objective value, which means the LP relaxation of (IP-M) has been solved to optimality.

Observe that only supported NDPs can be generated as columns in the LP relaxation of (IP-M): to obtain unsupported NDPs, which may be required in a solution to the original IP, it is necessary to restrict search in the MOP subproblems to regions "between" supported NDPs. Such regions are initialized in Phase 2, and explored and further refined, as needed, in Phase 3.

As an enhancement, the pricing problems are also used to obtain valid inequalities to tighten the lower bounding problem, (LB-M), which will be formed in the next phase. These inequalities are equivalent to the Lagrange cuts given in [11], and also discussed in [12], which is apparent from the well known correspondence between the column generation master LP and Lagrangian dual problem. Since we derive the cuts in the context of column generation, from the solution to the pricing problems, we refer to them as *pricing cuts*: if for some $i \in \mathcal{M}$ the solution to the pricing problem, (19) has value $\tilde{r}_i$, it must be that $z_{m+1} - \hat{\gamma}^\top z_{[m]} \geq \tilde{r}_i + \hat{\alpha}_i$ holds for any $z \in \mathcal{Z}^i$. As $\mathcal{N}^i \subseteq \mathcal{Z}^i$, this inequality is satisfied by all points in $\mathcal{N}^i$, and is a valid inequality for the NDF of MOP(i).

The pricing cuts are applied to the regions $\{\mathcal{Q}^i\}_{i,r}$ used in the lower bounding problem, (LB-M), since the points, $\mathcal{P}^i$ satisfy the pricing cuts by construction, whereas there might be points in a $\mathcal{Q}^i_r$ region violating them. Therefore we add a constraint to enforce the logic that if a region $r \in \{1, \ldots, q_i\}$ is selected in a feasible solution of (LB-M) then $w^i_r - \hat{\gamma}^\top u^i_r \geq \tilde{r}_i + \hat{\alpha}_i$ has to be satisfied for $(u^i_r, w^i_r) \in \mathcal{Q}^i_r \cap \mathcal{N}^i$. This logic can be modeled linearly, and added to the (LB-M) formulation, as

$$w^i_r - \hat{\gamma}^\top u^i_r \geq (\tilde{r}_i + \hat{\alpha}_i)\mu^i_r \tag{21}$$

using the binary variable μ^i_r. Note that (21) becomes redundant when the region is not selected, i.e., $\mu^i_r = 0$, since (17e) implies $u^i_r = \mathbf{0}$ and $w^i_r = 0$.

A useful feature of the pricing cuts, especially those added in the final round of the Pareto column generation procedure, are that they guarantee that the value of the LP relaxation of (LB-M), which we denote by ν^{LB}_{LP}, is at least as good a lower bound on the original IP value as the value of the LP relaxation of (IP-M), which we denote by ν^{IP}_{LP}.

Proposition 3 *Suppose that for some $\{\mathcal{P}^i\}_{i\in\mathcal{M}}$ the column generation stopping criterion for solving the LP relaxation of (IP-M) is met, with optimal LP dual variables $\hat{\gamma}$ and $\{\hat{\alpha}_i\}_{i\in\mathcal{M}}$. If (LB-M) is formed from $\{\mathcal{P}^i\}_{i\in\mathcal{M}}$ and any collection of regions, $\{\mathcal{Q}^i_r\}_{\substack{i\in\mathcal{M}\\ r=1,\ldots,q_i}}$, and the pricing cuts, (21), derived from the pricing problems at the final iteration of column generation, using $\hat{\gamma}$ and $\{\hat{\alpha}_i\}_{i\in\mathcal{M}}$, are included in (LB-M) for every region, then $\nu^{LB}_{LP} \geq \nu^{IP}_{LP}$.*

Proof of Proposition 3 follows from Theorem 1 in [11], the fact that the optimal dual variables at the completion of column generation provide the optimal Lagrangian variables for the corresponding Lagrangian dual problem ([30]), the observation that for all $i \in \mathcal{M}$ all points in $\mathcal{P}^i$ satisfy the pricing cuts, and from the convexity constraint (17c) for (LB-M). The latter two imply that if (λ, w, u) is feasible for (LB-M) then the vector

$$z^i := \sum_{k=1}^{|\mathcal{P}^i|} \lambda_k^i z^{i,k} + \sum_{r=1}^{q_i} (u_r^i, w_r^i)$$

satisfies $z_{m+1}^i - \hat{\gamma}^T z_{[m]}^i \geq \tilde{r}_i + \hat{\alpha}_i$, which is the pricing cut from the final column generation iteration, for all $i \in \mathcal{M}$. For the interested reader, a simpler, more direct proof is given in Appendix A.5.

4.2.2 Constructing an initial disjunctive relaxation (Phase 2)

Now that we have a set of known NDPs, $\mathcal{P}^i$, for each $i \in \mathcal{M}$, we use this information to decompose the part of the criterion space of the MOP(i) subproblem not dominated by any point in $\mathcal{P}^i$ into a set of polyhedral regions. The idea is to refine the criterion search space by eliminating the parts which are known to include no NDPs. Specifically, we will use the fact that if $z^* \in \mathcal{N}^i$, then the pointed cone obtained by adding the nonnegative orthant to z^* does not include any NDPs besides z^*, that is, $\left(\{z^*\} + \mathbb{R}_+^{m+1}\right) \cap \mathcal{N}^i = \{z^*\}$. Therefore, we can eliminate $\{z^*\} + \mathbb{R}_+^{m+1}$ from the search space. Then, for the MOP(i) subproblem, the criterion space that still needs to be considered for the remaining NDPs is

$$\mathbb{R}^{m+1} \setminus \bigcup_{k=1}^{|\mathcal{P}^i|} \left(\{z^{i,k}\} + \mathbb{R}_+^{m+1}\right) = \bigcap_{k=1}^{|\mathcal{P}^i|} \left(\bigcup_{j=1}^{m+1} \{z \in \mathbb{R}^{m+1} : z_j < z_j^{i,k}\} \right).$$

Given this observation, Algorithm 2 provides one simple way to create polyhedral regions, so that (14) is satisfied.

Algorithm 2 Decompose the criterion space of MOP(i), for $i \in \mathcal{M}$, into a set Λ^i of polyhedral regions, based on the set $\mathcal{P}^i$ of known NDPs

```
1. Initialize the list of regions $\Lambda^i = \left\{\{z \in \mathbb{R}^{m+1} : z \leq z^S(i)\}\right\}$
2. for all $z^* \in \mathcal{P}^i$
3.   for all $\mathcal{Q} \in \Lambda^i$
4.     if $z^* \in \mathcal{Q}$
5.       Let $\Lambda^i := (\Lambda^i \setminus \{\mathcal{Q}\}) \cup \bigcup_{j=1}^{m+1} \{\{z \in \mathcal{Q} : z_j \leq z_j^* - \epsilon\}\}$
6. Remove redundant regions in $\Lambda^i$
```

We note that the decomposition technique used in Algorithm 2 is the same as the so-called *full m-split* in [14] when all the initial lower and upper bounds are taken as negative infinity and the supernal point, respectively. As is customary in multiobjective optimization, ϵ is a small positive number that is used to express strict inequalities. Since we consider only pure integer problems with integer data, we can use $\epsilon = 1$. The algorithm starts with the initial criterion search region $\{z \in \mathbb{R}^{m+1} : z \leq z^S(i)\}$ where $z^S(i)$ denotes the supernal point of MOP(i), goes through all of the known NDPs one by one, and at every step replaces a region on the list with $m+1$ smaller regions if the region includes the NDP under consideration. As this process typically creates nested, and hence, *redundant*, regions (when $m \geq 2$), as suggested by [14], we detect and remove them from the decomposition at the last step. An illustration of initial region creation is provided in Appendix B.

Lastly, at the end of Phase 2, after decomposing the criterion space of each MOP(i) subproblem into NDPs and polyhedral regions, the lower bound problem (LB-M) is formed, including the pricing cuts saved in the previous step.

4.2.3 Refining the disjunctive relaxation (Phase 3)

In the main part of the algorithm, the lower (and upper) bound problem, (LB-M) (and (UB-M)), is iteratively tightened by generating more NDPs, refining the regions based on these new NDPs and adding more pricing cuts. The key idea is to make sure that at every iteration the lower bound problem makes progress, by refining some regions, which are selected based on the current solution of the (LB-M), so that the solution of (LB-M) is cut off. Also, the NDPs discovered during the refinement process may help the upper bound problem to make progress.

At every iteration, we first solve (LB-M). If $\mu = \mathbf{0}$ in the optimal solution, then the existing regions do not include any points that can improve the current lower bound. In this case, the lower and upper bound problems yield the same value, which must be optimal, so the algorithm can stop.

Consider an iteration at which $(\hat{\lambda}, \hat{\mu}, \hat{u}, \hat{w})$ is an optimal solution of (LB-M) and none of the stopping conditions are satisfied. Then there must be some $\hat{i} \in \mathcal{M}$ and $\hat{r} \in \{1, \ldots, q_i\}$ such that $\hat{\mu}^{\hat{i}}_{\hat{r}} = 1$, so region $\mathcal{Q}^{\hat{i}}_{\hat{r}}$ has been selected in the (LB-M) solution. We denote the underlying solution of (LB-M) in the criterion space of MOP($\hat{i}$) by $\hat{z}$, that is, $\hat{z} := (\hat{u}^{\hat{i}}_{\hat{r}}, \hat{w}^{\hat{i}}_{\hat{r}}) \in \mathcal{Q}^{\hat{i}}_{\hat{r}}$. In order to make sure that (LB-M) will make progress in the next iteration, the goal is to cut off its current solution. Specifically, we either prove that $\hat{z}$ is an NDP of MOP($\hat{i}$), in which case we add it to $\mathcal{P}^{\hat{i}}$, or refine $\mathcal{Q}^{\hat{i}}_{\hat{r}}$ in such a way that $\hat{z}$ does not belong to it anymore. Thus, to satisfy (14), we create a family of subregions of $\mathcal{Q}^{\hat{i}}_{\hat{r}}$, say $\Gamma = \{\mathcal{T}_1, \ldots, \mathcal{T}_{|\Gamma|}\}$, and a subset of the NDF of MOP($\hat{i}$), say $\mathcal{S}$, so that (i) all NDPs of MOP($\hat{i}$) that are in $\mathcal{Q}^{\hat{i}}_{\hat{r}}$ lie in $\mathcal{S}$ or in some set in Γ, and (ii) the family of subregions cuts off $\hat{z}$, i.e.,

$$\mathcal{Q}^{\hat{i}}_{\hat{r}} \cap \mathcal{N}^{\hat{i}} \subseteq \mathcal{S} \cup \bigcup_{\mathcal{T} \in \Gamma} \mathcal{T} \text{ and } \hat{z} \notin \bigcup_{\mathcal{T} \in \Gamma} \mathcal{T}. \tag{22}$$

Then, we make the updates

$$\mathcal{P}^{\hat{i}} = \mathcal{P}^{\hat{i}} \cup \mathcal{S} \text{ and } \Lambda^{\hat{i}} = (\Lambda^{\hat{i}} \setminus \{\mathcal{Q}^{\hat{i}}_{\hat{r}}\}) \bigcup \{\mathcal{T} \,:\, \mathcal{T} \in \Gamma\}$$

to improve (LB-M) and (UB-M). That is, we add the NDPs in $\mathcal{S}$ to (UB-M), while we replace $\mathcal{Q}^{\hat{i}}_{\hat{r}}$ in (LB-M) by the NDPs in $\mathcal{S}$ and the subsets in Γ. Also, we inherit all pricing cuts valid for $\mathcal{Q}^{\hat{i}}_{\hat{r}}$, for all $\mathcal{T} \in \Gamma$ and possibly generate new cuts for them as explained in detail next.

Algorithm 3 Given a region $\mathcal{Q}^{\hat{i}}_{\hat{r}}$, a point $\hat{z} \in \mathcal{Q}^{\hat{i}}_{\hat{r}}$ and a vector $\beta \in \mathbb{R}^{m+1}_{+}$, refine the region in such a way that (22) is satisfied

```
1.  Initialize $\mathcal{L} = \{\mathcal{Q}^{\hat{i}}_{\hat{r}}\}$, $\Gamma = \emptyset$ and $\mathcal{S} = \emptyset$.
2.  while $\mathcal{L} \neq \emptyset$          (or any early stopping criterion has been satisfied)
3.      Pick $\mathcal{T} \in \mathcal{L}$ and remove $\mathcal{T}$ from $\mathcal{L}$.
4.      Let $\nu^{\mathcal{T}} := \min \{\beta^\top z : z \in \mathcal{T} \cap \mathcal{Z}^{\hat{i}}\}$.
5.      if $\nu^{\mathcal{T}} < \infty$
6.          Save cut $\beta^\top z \geq \nu^{\mathcal{T}}$ for $\mathcal{T}$.
7.          Let $z(\mathcal{T}) \in \arg\min \{\beta^\top z : z \in \mathcal{T} \cap \mathcal{N}^{\hat{i}}\}$, add $z(\mathcal{T})$ to $\mathcal{S}$.
8.          if $\nu^{\mathcal{T}} > \beta^\top \hat{z}$
9.              for $j = 1, \ldots, m+1$
10.                 Add $\mathcal{T} \cap \{z : z_j \leq z_j(\mathcal{T}) - \epsilon\}$ to $\Gamma$, inheriting saved cuts for $\mathcal{T}$.
11.         else
12.             for $j = 1, \ldots, m+1$
13.                 if $z_j(\mathcal{T}) \leq \hat{z}_j$
14.                     Add $\mathcal{T} \cap \{z : z_j \leq z_j(\mathcal{T}) - \epsilon\}$ to $\Gamma$, inheriting saved cuts for $\mathcal{T}$.
15.                 else
16.                     Add $\mathcal{T} \cap \{z : z_j \leq z_j(\mathcal{T}) - \epsilon\}$ to $\mathcal{L}$, inheriting saved cuts for $\mathcal{T}$.
```

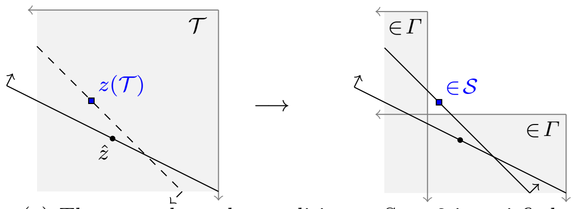
In Algorithm 3, we propose a general procedure that constructs the desired family Γ of subsets of $\mathcal{Q}^{\hat{i}}_{\hat{r}}$, together with the set $\mathcal{S}$. The set $\mathcal{L}$ represents a list of subsets of $\mathcal{Q}^{\hat{i}}_{\hat{r}}$ that have yet to be resolved, in the sense that they contain $\hat{z}$. At every iteration, we remove one element, $\mathcal{T} \subseteq \mathcal{Q}^{\hat{i}}_{\hat{r}}$, from $\mathcal{L}$ and possibly add new elements to Γ and/or $\mathcal{L}$. We use a given vector $\beta \in \mathbb{R}^{m+1}_{+}$ to search the region for unknown NDPs. Unless infeasible, the minimization problem at Step 4 provides an inequality, given at Step 6, that is valid for the NDPs in $\mathcal{T}$. Since the form of the optimization problem solved to identify them and the form of the inequalities themselves are identical to pricing cuts, we refer to these cuts, too, as *pricing cuts*. Such cuts are saved to be added to (LB-M) later. Moreover, if $\beta > \mathbf{0}$, then its optimal solution, say z^{Step4}, is guaranteed to be an NDP, thus can be directly added to $\mathcal{S}$ (i.e., we let $z(\mathcal{T}) = z^{Step4}$ at Step 7). Otherwise, an additional step, to search for an actual NDP, $z(\mathcal{T})$, on the hyperplane $\beta^\top z = \nu^{\mathcal{T}}$, is performed. Such an NDP exists whenever the problem at Step 4 is feasible, and can be found by lexicographic minimization of linear objectives over $\mathcal{T} \cap \mathcal{Z}^{\hat{i}}$ with additional upper bound constraints:

$$\operatorname{lex\,min}\ (z_1, \ldots, z_{m+1}) \text{ s.t. } z \in \mathcal{T} \cap \mathcal{Z}^{\hat{i}} \text{ and } z \leq z^{Step4} \tag{23}$$

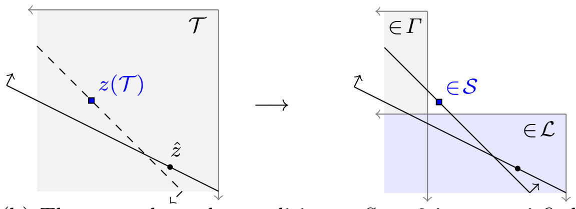
(where lex min is defined in Section 2).

We decompose $\mathcal{T}$ into $m+1$ smaller sets based on the newly found NDP. If the pricing cut found cuts off $\hat{z}$, then we add the subsets obtained to Γ (see

Figure 3(a)). Otherwise, we add those containing $\hat{z}$ to $\mathcal{L}$ to be processed later, and add the rest to $\varGamma$ (see Figure 3(b)). We note that due to the construction of the $\mathcal{T}$ sets, we always preserve the structure of MOP$(\hat{i})$. Thus we expect that the solution of (23) can be found with relative ease.



(a) The case where the condition at Step 8 is satisfied.



(b) The case where the condition at Step 8 is not satisfied.

Fig. 3 Illustration of an iteration of Algorithm 3 for two cases. On the left figures, the shaded area, the straight line, the dashed line represent the region to be refined ($\mathcal{T}$), a previously found cut, and the objective function used to search for new NDPs ($\beta^\top z$), respectively.

Note that each of the sets added to $\varGamma$ excludes $\hat{z}$. Hence, the (LB-M) problem is guaranteed to make progress as its optimal solution has been cut off. That this is sufficient to guarantee finite convergence of the algorithm is established in the next section.

4.3 Convergence of the algorithm

We first provide the following lemma.

Lemma 2 *Algorithm 3 converges finitely.*

Proof Observe that the generated cuts for a set of the form $\mathcal{T}$ in the above procedure are valid, in the sense that they cannot remove NDPs from $\mathcal{T}$, other than $z(\mathcal{T})$, which is added to $\mathcal{S}$. Also, observe that each time a set $\mathcal{T}$ is removed from $\mathcal{L}$, either no more sets are added to $\mathcal{L}$, or the sets that are added exclude $z(\mathcal{T})$, an NDP. Since there is a finite number of NDPs, the algorithm terminates in a finite number of iterations. $\square$

We may now prove that the algorithm terminates finitely.

Theorem 1 *Algorithm 1 with Step 3 (iii) (b) performed using Algorithm 3 converges finitely.*

Proof Each MOP subproblem has finitely many NDPs since its feasible region is bounded. At each iteration of Algorithm 1, when we refine a region by Algorithm 3, we either discard the region, or discover a new NDP from that region. Each NDP can be discovered at most once: in Algorithm 3 and in Algorithm 1, Step 3(iv) any NDPs discovered are removed from all regions by the refinement operation. Each NDP creates at most $m + 1$ subregions for each region it is contained in, so a finite number of regions are created in Steps 3(iii)(b) and 3(iv) of Algorithm 1. Therefore, Lemma 2 implies the finite convergence of Algorithm 1. □

4.4 Algorithm options: discussion

As already mentioned, the algorithm described above is only one specific incarnation within the MOP-based decomposition framework. Many algorithmic options are available. We discuss a some of them here.

There are different choices for the nonnegative vector, β, used to search a selected region for unknown NDPs. Below, we state a few options together with possible motivations.

1. *Any positive vector*: In the case that $\beta > \mathbf{0}$, if feasible, any optimal solution of the optimization problem at Step 4 of Algorithm 3 is guaranteed to be an NDP. Therefore, solving an additional lexicographic minimization problem at Step 7 would not be needed.
2. *Duals from (LB-M)*: After solving the (LB-M) problem, we fix all integer variables to their optimal values, and re-solve (LB-M) as an LP. Then, we let $\beta = (-\gamma^{LB}, 1)$ where γ^{LB} is the vector of optimal dual multipliers corresponding to the resource linking constraints, (17b), in this LP. In that case, $\nu^{\mathcal{T}} < \beta^{\top}\hat{z}$ can not happen as $\hat{z}$ has zero reduced cost, and any other point has nonnegative reduced cost. This might lead Algorithm 3 to terminate faster as it would be less likely to get into the case at Step 11, and thus the case at Step 16, where a new unresolved set is created.

Pricing cuts are optional, and may be used to varying degrees. They may be saved from any iteration of the column generation in Step 1 of Algorithm 1, and from any iteration of Algorithm 3. While their use can be expected to strengthen the lower bound obtained from solving (LB-M), they will also increase the size of the (LB-M) formulation, which may make it slower to solve; there is a trade-off to be considered.

A major source of alternatives is in the degree to which regions should be refined at each iteration. An aggressive approach to refining regions could be expected to cause rapid growth in the size of (LB-M), but fewer iterations (fewer times that (LB-M) needs to be solved). There is a trade-off between solving more, easier instances of (LB-M) (under a conservative refinement strategy) and solving fewer, harder instances (under an aggressive strategy).

Some of the algorithm options for region refinement available within the MOP-decomposition framework are as follows.

(I) Step 2 of Algorithm 1 is quite aggressive, with splitting for *every* block $i \in \mathcal{M}$ and *every* NDP found in Step 1. For the algorithm to make progress, it suffices to consider *one* block i and *one* NDP k for that block for which λ^i_k takes a fractional value in the LP relaxation of (IP-M). There are, of course, alternatives between these two extremes. For example, we may split (run Algorithm 2) with the input set of NDPs taken to be the set of $z^{i,k} \in \mathcal{P}^i$ for which λ^i_k takes a positive value in the optimal solution of the LP relaxation of (IP-M).

(II) The combination of Step 3(iii) of Algorithm 1 and Algorithm 3 lies at the conservative end of the spectrum: new NDPs are generated from one block for which a region is selected in the solution to (LB-M) and this region is explored thoroughly until the current solution is cut off. This process may produce many new NDPs and regions, depending on how many times the step in Lines 12–16 of Algorithm 3 is needed. If the test at Line 8 succeeds, then Algorithm 3 stops immediately after executing Lines 9–10, with the current (LB-M) solution cut off by the cut saved at Line 6. A more conservative approach would be to (provisionally) explore multiple blocks to find *one* needing the *fewest* new NDPs and regions to cut off the current solution, and use only that to update (LB-M). A more aggressive approach would be to explore *all* blocks for which a region is selected in the solution to (LB-M)and create new regions for all of them.

(III) Step 3(iv) of Algorithm 1 refines all regions for the block selected in Step 3(iii)(a) based on the new NDPs found in running Algorithm 3 for the selected region. Specifically, after refining the region for $\hat{i} \in \mathcal{M}$ selected in Step 3(iii)(a) using Algorithm 3 (run in Step 3(iii)(b), if $|\mathcal{S}| > 0$, then the NDPs in $\mathcal{S}$ are also used to refine other regions in the list $\Lambda^{\hat{i}}$. Thus, for any region in $\Lambda^{\hat{i}}$ that includes any of the newly generated NDPs[3], we run Algorithm 2, skipping its initialization step, with the input set $\mathcal{S}$ of NDPs instead of $\mathcal{P}^i$. A conservative alternative would be to refine only that one region, i.e., skip Step 3(iv). Of course, there are many strategies between these two extremes.

(IV) When pricing cuts are saved from within Algorithm 3 (at Line 6), if the test at Line 8 of the algorithm succeeds, the current (LB-M) solution has been cut off, and no further region refinement is needed. Thus a more conservative strategy than that stated in Lines 9–10 of Algorithm 3 would be to simply omit these lines of the algorithm and so rely only on the pricing cut saved at Line 6.

Finally, alternative stopping criteria are available, depending how the optional Step 3(ii) of Algorithm 1, solving (UB-M), is used. For example, (UB-M) can be solved every $\tau > 0$ iterations and the algorithm stopped if the optimal

[3] Recall that in the presence of at least two linking constraints, ($m \geq 2$), the regions for a block may overlap; they do not necessarily partition the MOP subproblem objective space. Thus an NDP found in one region may also be contained in another.

value of (UB-M) is close enough to the value of (LB-M). If (UB-M) is never used, stopping when $\mu = \mathbf{0}$, so (LB-M) has found an optimal solution, is the natural stopping criterion. Other stopping criteria, such as time, iteration or memory limit, may also be used, and (UB-M) solved once to seek a feasible solution, but there is no guarantee that (UB-M) will be feasible in this case, and so such stopping criteria make the algorithm an heuristic.

5 Numerical experiments

We first conduct preliminary experiments to see the effect of some algorithmic decisions mentioned in Section 4.4. Then, we compare the performance of our final algorithm with a generic branch-cut-and-price solver and a standard MILP solver.

We implement our algorithms in C++. For branch-cut-and-price, we use GCG 2.1.4 which is based on the branch-and-cut-and-price framework SCIP 5.0.1. In all algorithms, we use IBM ILOG CPLEX 12.8 as the LP/MILP solver. We perform all experiments using a single thread on a Mac OS X 10.15 with 2.2 GHz Intel Core i7 CPUs and 16 GB RAM. In all the experiments, we use a solution time limit of one hour.

Our implementation of Algorithm 1 includes the redundancy check to detect and remove redundant regions, at every iteration, as this was found to be crucial to efficiency. Details of computational experiments suppporting this conclusion are given in Appendix D.1. Our implementation does not solve the upper bound problem, (UB-M), instead terminating when (LB-M) yields a feasible solution (or the time limit is reached). In our implementation, the block selected at Step 3(iii)(a) of Algorithm 1 is that with smallest index i for which a region, not an NDP, has been selected in the solution to (LB-M).

5.1 Test instances

We consider two groups of instances.

Our first test set consists of randomly generated instances of IPs with two coupling constraints, three to ten subproblems with knapsack structure, and different right-hand-side values of the coupling constraints. More specifically, we first generate M-many triobjective 3-dimensional knapsack cover (KC) problems with $n = 20$ items for the MOP subproblems, and then link their resources with two capacity constraints. That is, we have MOP(i) : $\min\{A_1^i x, A_2^i x, (c^i)^\top x^i\}$ s.t. $x \in \mathcal{X}_{KP}^i = \left\{x \in \{0,1\}^{20} : \sum_{j=1}^{20} G_{kj}^i x_j \geq g_k^i,\ k = 1,2,3\right\}$ for some $G^i \in \mathbb{R}_+^{3\times 20}$ and $g^i \in \mathbb{R}_+^3$. The objective function and knapsack constraint coefficients, i.e., the components of A_1^i, A_2^i, c^i and G^i, are drawn uniformly from the set $\{0, 1, \ldots, 100\}$. Then, the right-hand sides of the knapsack constraints are set as $g_k^i = \sum_{j=1}^{20} G_{kj}^i/2$ for all $k = 1, 2, 3$. We consider different sizes of M, namely $M \in \{3, 4, \ldots, 10\}$, and instances with different values of the resource vector b. We label an instance as "M_b_1, b_2". For this class of problems, CPLEX very easily detects infeasible instances and easily solves instances with looser coupling constraints. Therefore, for each instance, we tried several alternative resource vectors, b, and retained instances whose resource vector entries just large enough to be feasible.

For the second data set, we consider the multidimensional multiple-choice knapsack problem (MMKP), one of the hardest variants of the knapsack problem, having many real-world applications [20]:

$$\max \left\{ \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}} p_{ij}x_{ij} : \sum_{j\in\mathcal{J}} x_{ij} = 1, i \in \mathcal{M}, \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}} w_{ijk}x_{ij} \le b_k, k \in \mathcal{K} \right\}.$$

We build our test data set based on the well-known MMKP benchmark instances[4], originally proposed in [25]. Specifically, we use the regular structure instances INST01-INST20, where the number of groups (i.e., blocks in our decomposition), $|\mathcal{M}|$, ranges from 50 to 350, and the number of items in a group, $|\mathcal{J}|$, ranges from 10 to 30. All these instances have dimensionality of 10, which we reduce to make them loosely coupled. Namely, we consider $|\mathcal{K}| \in \{3, 4, 5\}$, by taking the first $|\mathcal{K}|$ linking constraints (out of 10) in the original data. We note that based on numerical experiments, the instances can be classified into two groups as INST01-INST12 and INST13-INST20 in terms of difficulty. Since their difficulty corresponds to their sizes, in what follows we refer to them as small and large instances, respectively.

5.2 Experiments on algorithm options

In what follows, we investigate the impact of some alternative algorithm options on the algorithm performance. As discussed in Section 4.4, there are many such options. Here we select two settings in each of the key areas offering choice and investigate all combinations of these. Specifically, we test all combinations of: (i) searching regions using the constant $\beta = \mathbf{1}$ vs β supplied by the dual from (LB-M); (ii) adding all pricing cuts vs adding none; and (iii) two region refinements strategies, one towards the conservative and the other towards the aggressive end of the spectrum. Details of (i) and (iii) are given in the next section, with conclusions from the experimental results summarized in the section after.

5.2.1 Details of alternatives tested

β vector selection. We consider the two options stated at the end of Section 4.4 for the vector β used in Algorithm 3 to search regions for NDPs, namely $\beta = \mathbf{1}$, and $\beta = (-\gamma^{LB}, 1)$. For the latter, we test three fallback strategies when the dual multiplier vector γ^{LB} has a zero element, which might lead to

[4] http://www.info.univ-angers.fr/pub/hao/mmkp.html

the discovery of weakly nondominated points: switch to $\beta = \mathbf{1}$ ("All One"), add a small positive value to zero components ("Small Pos"), and leave it as is ("Leave As Is").

Region refinement. We compare the region refinement steps described in Section 4.2.2 and Section 4.2.3, which we refer to as the Aggressive ("Agg") strategy, with an alternative that aims to create fewer regions, which we refer to as the Conservative ("Con") strategy. The latter is defined by the following modifications to the algorithm.

(a) In Step 2 of Algorithm 1, use the last alternative discussed in (I), in Section 4.4: for each block $i \in \mathcal{M}$, run Algorithm 2 with the input set of NDPs as those selected with positive value in the optimal solution of the LP relaxation of (IP-M).
(b) In Step 3 of Algorithm 1, adopt the most conservative strategy discussed in (III), in Section 4.4: skip Step 3(iv) of Algorithm 1.
(c) Adopt the most conservative strategy discussed in (IV), in Section 4.4: when pricing cuts are saved, skip Lines 9–10 of Algorithm 3.

5.2.2 Experimental results on alternatives

Solution times for all combinations of alternatives for choice of β, use of pricing cuts and region refinement strategy, on all the KC instances and the easiest MMKP instances, namely the first 12 MMKP instances with $|\mathcal{K}| = 3$, are presented in Table 5 of Appendix D.2. We draw the following conclusions.

Pricing cuts are helpful only for easy instances; for more time-consuming instances (say the ones taking more than 15 seconds to solve), they create a significant overhead for the lower bounding problem, increasing the solution times drastically (usually two orders of magnitude for the MMKP instances).

The choice of fallback strategy for β supplied by the (LB-M) dual values seems to make little difference to algorithm performance. The use of a constant $\beta = \mathbf{1}$ throughout the algorithm is slightly worse, overall, than getting guidance from the (LB-M) duals for the KC instances, but slightly better for the MMKP instances, which are by far the harder of the two classes, on average. Further insights about this are given in Appendix D.3.

To determine the better choice of β overall, we provide a supplementary table of results, Table 6 in Appendix D.3, for somewhat harder small MMKP instances, (having five linking constraints instead of three), focused on the setting with no pricing cuts. This shows that $\beta = \mathbf{1}$ throughout is consistently better than the alternatives tested.

The Aggressive region refinement strategy is helpful for easy instances, reducing solution times by 22% and 30% on the average for the KC and small MMKP instances, respectively, compared to the Conservative strategy. However, as we expect the Aggressive strategy to become increasingly burdensome as instance size and difficulty increase, we present results for harder small MMKP instances, with five linking constraints instead of three, in Table 6 of

Appendix D.2. (Here we only show results for the setting with no pricing cuts, which is by orders of magnitude the superior alternative for MMKP instances.) These show the benefit of the Conservative strategy, which reduces the solution times by 32% on average compared to those of the Aggressive strategy, despite doubling the number of iterations (seen in Table 8 of Appendix D.4).

5.3 Performance

In this section, we compare our MOP-based decomposition algorithm ("MOP dec") with a standard MILP solver, CPLEX, and a generic branch-cut-and-price solver for MILPs, GCG, both with default settings except we use a single thread. (For GCG, we explicitly provide the block structure and linking constraints.) The MOP-based decomposition algorithm uses the settings found to be best overall, as discussed in the previous section: no pricing cuts, constant $\beta = \mathbf{1}$ throughout and the Conservative refinement strategy.

Comparison on KC instances. The comparison results for the KC instances are given in Table 1, where the optimal objective values and solution times (in seconds) are in the columns labeled as "Opt" and "Time", respectively. The best of the CPU times are displayed in bold. For CPLEX, we also report lower bounds (LB) and upper bounds (UB) obtained at the end, together with the number of branch-and-bound nodes executed in total (Nd). The latter is provided for GCG as well.

	MOP dec		CPLEX				GCG	
Instance	Opt	Time	Time	LB	UB	Nd	Time	Nd
3_850,950	1041	14	**0**	1041	1041	1823	24	126
3_900,950	959	14	**0**	959	959	233	9	27
3_950,950	937	14	**0**	937	937	1087	8	39
4_1200,1200	1298	16	**0**	1298	1298	2374	31	103
4_1300,1300	1078	15	**0**	1078	1078	2645	5	9
4_1400,1400	1014	16	**0**	1014	1014	456	3	5
5_1500,1600	1592	**21**	176	1592	1592	> 2M	26	79
5_1600,1500	1725	**20**	373	1725	1725	> 6M	24	69
5_1600,1550	1544	20	**1**	1544	1544	1693	25	77
6_1820,1900	2154	**24**	1011	2154	2154	> 11M	27	73
6_1900,1900	1849	22	**1**	1849	1849	1368	45	137
6_1900,1850	2007	**23**	2979	2007	1985	> 37M	56	178
7_2250,2250	2477	28	**1**	2477	2477	1066	54	139
8_2600,2500	3213	**30**	3600	3094	$+\infty$	> 18M	98	292
9_2800,2900	3666	**34**	3600	3441	$+\infty$	> 10M	176	475
10_3100,3150	3763	**39**	3600	3685	$+\infty$	> 1M	198	506

Table 1 Comparison with CPLEX and GCG for KC instances. The settings used for MOP dec are: no pricing cuts, $\beta = \mathbf{1}$, and conservative region refinement.

The MOP-based decomposition algorithm solves all instances in less than one minute. Moreover, in terms of solution times, it scales well with the number of blocks. On the other hand, although CPLEX solves the instances with a smaller number of blocks much faster, it spends a significantly larger amount of time for the instances with a large number of blocks. In particular, it cannot solve the last three instances, even fails to find any feasible solution, in one hour. Also, in many test instances, it processes more than a million nodes. While GCG is a viable approach, it usually takes more time (88% more on average over all the instances) to converge, and is increasingly worse as instance difficulty increases.

Comparison on MMKP instances. Table 2 provides the performance comparison for all the MMKP instances with four and five linking constraint cases. CPLEX solves all the instances quickly, the longest taking around three minutes, while the MOP-based decomposition algorithm solves all, with the longest taking around 15 minutes. On the other hand, GCG struggles to prove optimality, and hits the time limit for all the large instances and some small instances with five linking constraints.

Discussion of Phase 1. We note that Phase 1 of Algorithm 1, i.e., solving the LP relaxation of (IP-M), usually takes a small amount of the total time. More specifically, for the KC instances, it constitutes 1.9% of the total time. For the small MMKP instances, that portion is 32.0% for the $|\mathcal{K}| = 4$ instances and 6.5% for $|\mathcal{K}| = 5$ instances. For the large MMKP instances, it is 3.4% for the $|\mathcal{K}| = 4$ instances and , 1.1% for the $|\mathcal{K}| = 5$ instances. The only seemingly large percentage is 32.0% for the small MMKP instances with $|\mathcal{K}| = 4$. However, those instances took only 2.1–4.2 seconds to solve. On the other hand, GCG spends 1.8%, 8.9% and 5.5% of the total solution time (or of one hour if the time limit is hit) for the KC, small MMKP and large MMKP instances, respectively, at the root node.

Lower bound improvement rate comparison. Lastly, we compare the lower bounds obtained in the course of decomposition algorithms. In Figure 4, we illustrate the lower bound improvement over time for a KC instance for GCG and four variants of the MOP-based decomposition algorithm. All four use constant $\beta = \mathbf{1}$ throughout. The "Base" setting has no pricing cuts and Conservative region refinement. The other three variants are formed by adding pricing cuts, changing Conservative to Aggressive refinement, and doing both. In Figure 5, we present the evolution of the lower bound for an MMKP instance where the objective is flipped to minimization, thus the bounds are negative. In both figures, the x-axis and y-axis show the time in seconds and the lower bound value, respectively.

In Figure 4, we observe (for the KC instance) that both the pricing cuts and the Aggressive region refinement strategy improve the lower bound faster. Most notably, the pricing cuts added at the fist iteration (in 0.5 seconds) lead to a lower bound that is only 2.9% away from the optimal value. However,

				MOP dec		CPLEX		GCG	
$\vert\mathcal{K}\vert$	Instance	$\vert\mathcal{M}\vert$	$\vert\mathcal{J}\vert$	Opt	Time	Time	Nd	Time	Nd
4	INST01	50	10	12369	2.1	0.5	1510	315.6	3041
	INST02	50	10	13771	2.1	0.1	0	11.3	38
	INST03	60	10	13803	2.6	0.1	24	38.8	233
	INST04	70	10	17565	3.3	0.6	3150	488.6	3340
	INST05	75	10	19200	3.2	0.5	1650	154.5	906
	INST06	75	10	19092	3.2	0.6	2476	217.4	1322
	INST07	80	10	19474	3.7	0.5	2766	304.0	1786
	INST08	80	10	20089	3.6	0.5	1914	144.9	726
	INST09	80	10	20157	3.5	0.6	3351	1400.3	7986
	INST10	90	10	22505	3.8	0.3	502	300.2	1527
	INST11	90	10	22580	3.8	1.0	3217	543.7	2894
	INST12	100	10	24579	4.2	0.4	2229	1787.9	8877
	INST13	100	30	22538	96.6	8.1	28370	TL(0.02%)	23354
	INST14	150	30	34471	113.6	5.3	12679	TL(0.01%)	13895
	INST15	180	30	41000	133.6	18.6	108693	TL(0.01%)	14335
	INST16	200	30	45288	172.2	3.0	3719	TL(0.01%)	12077
	INST17	250	30	56690	159.8	12.4	33114	TL(0.01%)	9620
	INST18	280	20	62930	117.7	4.9	4659	TL(0.01%)	10462
	INST19	300	20	67730	87.4	13.9	21299	TL(0.01%)	9208
	INST20	350	20	79108	112.6	2.3	1594	TL(0.01%)	8524
5	INST01	50	10	11988	13.8	2.2	8851	1774.9	16060
	INST02	50	10	13771	8.4	0.1	0	11.3	36
	INST03	60	10	13200	13.3	0.9	2095	383.9	3083
	INST04	70	10	17081	17.6	5.9	24296	388.3	2668
	INST05	75	10	18883	20.1	8.7	38009	TL(0.03%)	26044
	INST06	75	10	18755	22.0	8.0	31703	TL(0.01%)	24527
	INST07	80	10	19003	21.9	2.1	7648	2029.6	13628
	INST08	80	10	19720	22.0	7.5	34758	TL(0.03%)	26348
	INST09	80	10	19804	18.5	4.9	17839	TL(0.01%)	25706
	INST10	90	10	22029	19.1	4.7	22031	TL(0.02%)	22594
	INST11	90	10	22126	17.2	3.1	8668	2965.2	19391
	INST12	100	10	24251	18.8	5.0	12579	TL(0.04%)	19444
	INST13	100	30	22283	911.4	144.3	722630	TL(0.03%)	21385
	INST14	150	30	34135	495.0	10.7	56464	TL(0.02%)	14301
	INST15	180	30	40541	578.2	56.6	201771	TL(0.02%)	11566
	INST16	200	30	44810	815.6	187.6	750325	TL(0.01%)	11533
	INST17	250	30	56075	567.9	22.1	24847	TL(0.01%)	9410
	INST18	280	20	62247	265.5	80.6	197777	TL(0.01%)	10634
	INST19	300	20	66960	246.0	68.8	189924	TL(0.01%)	9359
	INST20	350	20	77960	278.1	14.7	10946	TL(0.01%)	6811

Table 2 Comparison with CPLEX and GCG for MMKP instances. The settings used for MOP dec are: no pricing cuts, $\beta = \mathbf{1}$, and conservative region refinement.

without the Aggressive refinement strategy, the algorithm spends significantly more time (215 seconds versus 56 seconds) to close the remaining gap. On the other hand, GCG spends 2.4 seconds to reach to a lower bound that is 1.7% away from the optimal value and also spends a long time (196 seconds) to close the remaining gap.

In Figure 5, we observe (for the MMKP instance) that despite the lack of pricing cuts, (LB-M) provides significantly better lower bounds than GCG.

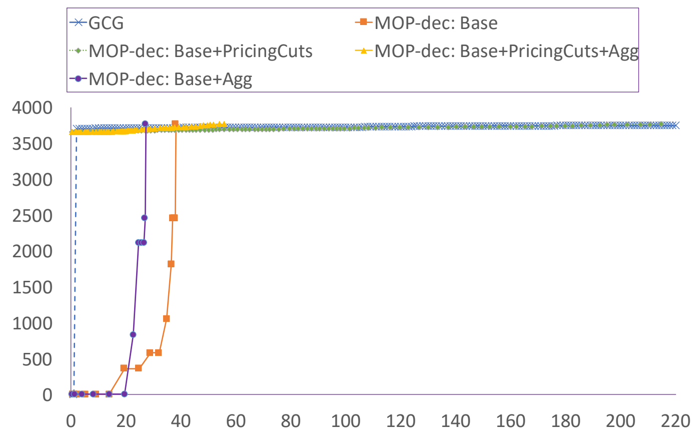


Fig. 4 Lower bound improvement over time for the KC instance 10_3100,3150, where x-axis and y-axis show the time in seconds and the lower bound value, respectively.

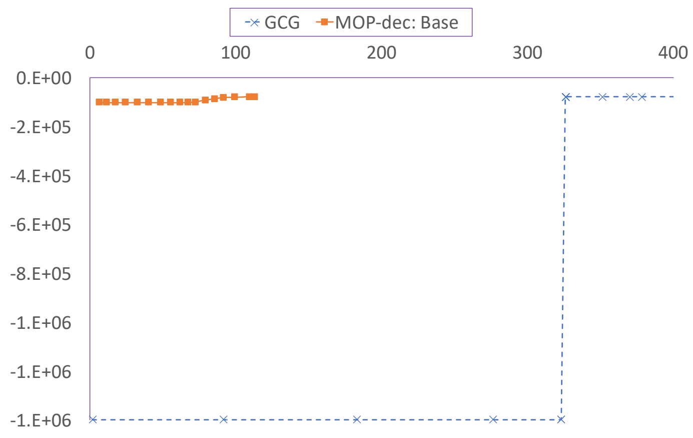


Fig. 5 Lower bound improvement over time for the MMKP instance INST20 with $|\mathcal{K}| = 4$, where x-axis and y-axis show the time in seconds and the lower bound value (for the minimization version), respectively.

The latter takes 323 seconds to reach to the lower bound obtained at the first iteration of the MOP-based decomposition algorithm in only 6 seconds.

6 Concluding remarks

We have presented a novel perspective, namely a new formulation and decomposition algorithm, to solve loosely coupled IPs. The formulation has an IP master lower bounding problem that is most naturally expressed as a disjunctive program, and a set of MOP subproblems which provide columns to the master problem.

From this perspective, we have developed a specific algorithm and, using a small set of random test instances, we have shown that our algorithm may be practical, since, for several instances, it finds an optimal solution in minutes, whereas a standard IP solver cannot find a feasible solution in an hour. However, further research is needed to achieve an efficient, general-purpose version of the algorithm, for example, by incorporating preprocessing, warm-starting, reduced-cost fixing and constraint/block aggregation ideas.

We have articulated the deep connections of the MOP-based reformulation with the value function reformulation, and explained the key differences. We have discussed similarities and differences between our framework and traditional branch-and-price, and how they might interconnect. This discussion shows the potential for new research at the intersection of branch-and-price, value function reformulation and disjunctive programming. Indeed, we are not aware of any prior work that takes a disjunctive programming approach to problems in which the terms of the disjunction are not explicitly available, and need to be generated dynamically, as is the case here for the lower bound problem in MOP-based decomposition. It makes clear that decomposition algorithms of the future could better hybridize techniques and better exploit trade-offs between efficacy of branching rules, difficulty of subproblems and difficulty of master problems, especially so as to more directly exploit current IP solver technology. Many new research opportunities offer themselves in this topic.

Acknowledgements The research was supported in part by the Office of Naval Research grant N00014-15-1-2078 and by the National Science Foundation EAGER grant CCF-1415460.

References

1. Ahmed, S., Tawarmalani, M., Sahinidis, N.: A finite branch and bound algorithm for two-stage stochastic integer programs. Math. Program. **100**(2), 355–377 (2004)
2. Balas, E.: Disjunctive programming. Annals of discrete mathematics **5**, 3–51 (1979)
3. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. Discrete Applied Mathematics **89**(1-3), 3–44 (1998)
4. Balas, E.: Disjunctive programming. Springer (2018)
5. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Mathematical programming **58**(1-3), 295–324 (1993)
6. Balas, E., Margot, F.: Generalized intersection cuts and a new cut generating paradigm. Mathematical Programming **137**(1-2), 19–35 (2013)
7. Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P.: Branch-and-price: Column generation for solving huge integer programs. Oper. Res. **46**(3), 316–329 (1998)
8. Bergman, D., Bodur, M., Cardonha, C., Cire, A.: Network models for multiobjective discrete optimization (2018). http://www.optimization-online.org/DB_FILE/2018/02/6483.pdf
9. Boland, N., Charkhgard, H., Savelsbergh, M.: A criterion space search algorithm for biobjective integer programming: The balanced box method. INFORMS J. Comput. **27**(4), 735–754 (2015)
10. Boland, N., Charkhgard, H., Savelsbergh, M.: The L-shape search method for triobjective integer programming. Math. Program. Comput. **8**(2), 217–251 (2016)

11. Boyd, E.A.: The Lagrangian and other primal cutting planes for linear integer programming problems. Technical Report TR90-3, Rice University, Department of Mathematical Sciences (1990)
12. Boyd, E.A.: Fenchel cutting planes for integer programs. Operations Research **42**(1), 53–64 (1994)
13. Chen, Q., Grossmann, I.: Modern modeling paradigms using generalized disjunctive programming. Processes **7**(11), 839 (2019)
14. Dächert, K., Klamroth, K.: A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. J. Global. Opt. pp. 1–34 (2015)
15. Dantzig, G., Wolfe, P.: Decomposition principle for linear programs. Oper. Res. **8**(1), 101–111 (1960)
16. Desaulniers, G., Desrosiers, J., Solomon, M.: Column generation, vol. 5. Springer Science & Business Media (2006)
17. Ehrgott, M.: Multicriteria optimization. Springer Science & Business Media (2006)
18. Ehrgott, M., Gandibleux, X., Przybylski, A.: Exact methods for multi-objective combinatorial optimisation. In: Multiple Criteria Decision Analysis, 2nd edn., pp. 817–850. Springer (2016)
19. Floudas, C., Pardalos, P.: Encyclopedia of optimization, vol. 1. Springer Science & Business Media (2008)
20. Gao, C., Lu, G., Yao, X., Li, J.: An iterative pseudo-gap enumeration approach for the multidimensional multiple-choice knapsack problem. European J. Oper. Res. **260**(1), 1–11 (2017)
21. Geoffrion, A.: Lagrangian relaxation for integer programming. In: 50 Years of Integer Programming 1958-2008, pp. 243–281. Springer (2010)
22. Grossmann, I.E., Trespalacios, F.: Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. AIChE Journal **59**(9), 3276–3295 (2013)
23. Jozefowiez, N., Laporte, G., Semet, F.: A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. INFORMS J. Comput. **24**(4), 554–564 (2012)
24. Kazachkov, A.M.: Non-recursive cut generation. Ph.D. thesis, Carnegie-Mellon University (2018)
25. Khan, S., Li, K., Manning, E., Akbar, M.: Solving the knapsack problem for adaptive multimedia systems. Stud. Inform. Univ. **2**(1), 157–178 (2002)
26. Kong, N., Schaefer, A., Hunsaker, B.: Two-stage integer programs with stochastic right-hand sides: A superadditive dual approach. Math. Program. **108**(2), 275–296 (2006)
27. Molina, F.: A survey of resource directive decomposition in mathematical programming. ACM Comput. Surv. **11**(2), 95–104 (1979)
28. Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization **19**, 79–102 (2016)
29. Nemhauser, G.: Decomposition of linear programs by dynamic programming. Naval Res. Logist. Quart. **11**(2), 191–195 (1964)
30. Nemhauser, G.L., Wolsey, L.A.: Integer and combinatorial optimization. John Wiley & Sons (1988)
31. Özlen, M., Burton, B.A., MacRae, C.A.G.: Multi-objective integer programming: An improved recursive algorithm. J. Optim. Theory Appl. **160**(2), 470–482 (2014)
32. Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F.: Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS J. Comput. **30**(2), 339–360 (2018)
33. Przybylski, A., Gandibleux, X., Ehrgott, M.: A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. Discrete Optimization **7**(3), 149–165 (2010)
34. Savelsbergh, M.: A branch-and-price algorithm for the generalized assignment problem. Operations research **45**(6), 831–841 (1997)
35. Sawaya, N.W., Grossmann, I.E.: A cutting plane method for solving linear generalized disjunctive programming problems. Computers & chemical engineering **29**(9), 1891–1913 (2005)

36. Soylu, B.: The search-and-remove algorithm for biobjective mixed-integer linear programming problems. European Journal of Operational Research **268**(1), 281–299 (2018)
37. Trapp, A., Prokopyev, O., Schaefer, A.: On a level-set characterization of the value function of an integer program and its application to stochastic programming. Oper. Res. **61**(2), 498–511 (2013)
38. Trespalacios, F., Grossmann, I.E.: Review of mixed-integer nonlinear and generalized disjunctive programming methods. Chemie Ingenieur Technik **86**(7), 991–1012 (2014)
39. Trespalacios, F., Grossmann, I.E.: Chapter 24: Review of mixed-integer nonlinear optimization and generalized disjunctive programming applications in process systems engineering. Advances and Trends in Optimization with Engineering Applications pp. 315–329 (2017)
40. Vanderbeck, F.: Branching in branch-and-price: a generic scheme. Math. Program. **130**(2), 249–294 (2011)
41. Vanderbeck, F., Savelsbergh, M.: A generic view of Dantzig-Wolfe decomposition in mixed integer programming. Oper. Res. Lett. **34**(3), 296–306 (2006)
42. Zhang, W., Reimann, M.: A simple augmented ε-constraint method for multi-objective mathematical integer programming problems. European J. Oper. Res. **234**(1), 15 – 24 (2014)

A Proofs

A.1 Proof of Lemma 1

Let $z \in \mathbb{R}^J$ be an NDP of (13), and, for contradiction, assume that z is not an NDP of (4). As z is clearly feasible for (4) and assumed not to be an NDP, there must exist $x' \in \mathcal{X}$ with $z' = (g_1(x'), \dots, g_J(x')) \leq z$ and $z' \neq z$. Then, combining this with (13c), we obtain $z'_j = g_j(x') \leq z_j \leq \delta_j$ for all $j \in \mathcal{J}'$. This implies that z' is also feasible for (13), which contradicts the fact that z is an NDP of (13). □

A.2 Proof of Proposition 1

Let $\{u^i\}_{i\in\mathcal{M}}$ be an arbitrary optimal solution of RDMP, and, for each $i \in \mathcal{M}$, let $\hat{x}^i$ be an optimal solution of the ith subproblem RDSP(i, u). Without loss of generality, we may take $\hat{x}^i$ to be an optimal solution such that $A^i\hat{x}^i$ is not dominated by $A^i x^i$ for any other optimal solution x^i, for example, take $\hat{x}^i$ to be an optimal solution of

$$\operatorname*{lex\,min}_{x\in\mathcal{X}^i: A^i x\leq u^i} ((c^i)^\top x, A^i_1 x, \dots, A^i_m x).$$

Let $\hat{u}^i := A^i\hat{x}^i$ for all $i \in \mathcal{M}$. Then, $f_i(\hat{u}^i) = (c^i)^\top\hat{x}^i = f_i(u^i)$, and $(\hat{u}^i, f_i(\hat{u}^i))$ is the criterion space image of $\hat{x}^i$, thus a feasible solution to MOP(i). Also, by construction of $\hat{x}^i$ and Lemma 1, it must be that $(\hat{u}^i, f_i(\hat{u}^i))$ is an NDP for MOP(i). Lastly, we observe that $\{\hat{u}^i\}_{i\in\mathcal{M}}$ is feasible for RDMP since

$$\sum_{i\in\mathcal{M}} \hat{u}^i = \sum_{i\in\mathcal{M}} A^i\hat{x}^i \leq \sum_{i\in\mathcal{M}} u^i \leq b,$$

and has the same objective value as that of $\{u^i\}_{i\in\mathcal{M}}$ as

$$\sum_{i\in\mathcal{M}} f_i(\hat{u}^i) = \sum_{i\in\mathcal{M}} (c^i)^\top\hat{x}^i = \sum_{i\in\mathcal{M}} f_i(u^i),$$

which completes the proof. □

A.3 Relationship with Value Function Reformulation

To establish the relationship between our multiobjective reformulation and the value function reformulation discussed in [37], we first adapt the definition of a *level-set minimal* vector given as Definition 1 in [37] to the context of the RDSP(i, u): the direction of the defining inequality is reversed, since [37] have a maximization problem, whereas RDSP(i, u) is a minimization problem.

Definition 1 Without loss of generality, assume that for given $i \in \mathcal{M}$ all entries of A^i are integer. Then $\hat{u} \in \mathbb{Z}^{n_i}$ is *level-set minimal for RDSP(i, u)* if $f_i(\hat{u} - \mathbf{e}_h) > f_i(\hat{u})$ for all $h = 1, \dots, n_i$, where $\mathbf{e}_h$ is the hth unit vector.

Proposition 4 *The pair $(u^i, f_i(u^i)) \in \mathcal{N}^i$ if and only if u^i is level-set minimal for RDSP(i, u).*

Proof Suppose first that u^i is *not* level-set minimal for RDSP(i, u). Then for some h it must be that $f_i(u^i - \mathbf{e}_h) \leq f_i(u^i)$. So there must exist an optimal solution, y say, of RDSP$(i, u^i - \mathbf{e}_h)$, with $f_i(u^i - \mathbf{e}_h) = (c^i)^T y$. By feasibility of y, $A^i y \leq u^i - \mathbf{e}_h \leq u^i$ with $u^i - \mathbf{e}_h \neq u^i$ and hence $A^i y \neq u^i$. Thus $(A^i y, (c^i)^T y)$ dominates $(u^i, f_i(u^i))$, and so $(u^i, f_i(u^i)) \notin \mathcal{N}^i$.

Now suppose that $(u^i, f_i(u^i)) \notin \mathcal{N}^i$, where $u^i \in \mathbb{Z}^{n_i}$. Then there exists $\hat{u}$ with $(\hat{u}, f_i(\hat{u})) \in \mathcal{N}^i$ and $(\hat{u}, f_i(\hat{u})) \leq (u^i, f_i(u^i))$. Hence $\hat{u} \neq u^i$. Since A^i is integer (and $\mathcal{X}^i \subseteq \mathbb{Z}^{n_i}_+$), $\hat{u} \in \mathbb{Z}^{n_i}$ so $\hat{u} \leq u^i - \mathbf{e}_h$ for some $h = 1, \dots, n_i$. Since the value function f_i is monotonically non-increasing, it must be that $f_i(\hat{u}) \geq f_i(u^i - \mathbf{e}_h)$. As $f_i(u^i) \geq f_i(\hat{u})$, we have that $f_i(u^i) \geq f_i(u^i - \mathbf{e}_h)$ and so u^i is not level-set minimal. □

A.4 Proof of Proposition 2

Given an optimal solution $\tilde{\lambda}$ of (7), let k_i be the index such that $\tilde{\lambda}^i_{k_i} = 1$ for $i \in \mathcal{M}$. Now, we will construct a feasible solution $(\hat{\lambda}, \hat{\mu}, \hat{u}, \hat{w})$ to (17). For any $i \in \mathcal{M}$, if $k_i \leq |\mathcal{P}^i|$, let $\hat{\lambda}^i_{k_i} = 1$, otherwise let $\hat{\mu}^i_{r_i} = 1$, where $r_i \in \{1, \dots, q_i\}$ is an arbitrary index of a region that includes the nondominated point z^{i,k_i}. (Such a region exists, by (14).) In the latter case, we also let $\hat{u}^{i,r_i} = z^{i,k_i}_{[m]}$ and $\hat{w}^i_{r_i} = z^{i,k_i}_{m+1}$. We set all other variable values to zero. Then, (17b)-(17g) hold and the objective function (17a) takes value ν^{IP} at $(\hat{\lambda}, \hat{\mu}, \hat{u}, \hat{w})$. □

A.5 Proof of Proposition 3

Given the conditions of the proposition, we will construct a feasible solution to the LP dual of a relaxation of the LP relaxation of (LB-M), with value equal to ν^{IP}_{LP}. Since $\hat{\gamma}$ and $\{\hat{\alpha_i}\}_{i \in \mathcal{M}}$ are the optimal dual variables for the LP relaxation of (IP-M), it must be that

$$z_{m+1} - \hat{\gamma}^T z_{[m]} - \hat{\alpha_i} \geq 0, \quad \forall z \in \mathcal{Z}^i, \ \forall i \in \mathcal{M}. \tag{24}$$

Thus the reduced cost of each pricing problem in the final column generation iteration is

$$\tilde{r}_i := \min\{z_{m+1} - \hat{\gamma}^T z_{[m]} - \hat{\alpha_i} \ : \ z \in \mathcal{Z}^i\} \geq 0, \qquad \forall i \in \mathcal{M}. \tag{25}$$

Furthermore, the LP relaxation of (IP-M) has objective value given by the optimal LP duals according to:

$$\nu^{IP}_{LP} = b^T \hat{\gamma} + \mathbf{1}^T \hat{\alpha_i}. \tag{26}$$

Now consider the following relaxation for the LP relaxation of (LB-M) with the addition of the pricing cuts from the final column generation iteration:

$$\nu^{LB}_R := \min \sum_{i \in \mathcal{M}} \left(\sum_{k=1}^{|\mathcal{P}^i|} \lambda^i_k z^{i,k}_{m+1} + \sum_{r=1}^{q_i} w^i_r \right) \tag{27a}$$

$$\text{s.t.} \sum_{i \in \mathcal{M}} \left(\sum_{k=1}^{|\mathcal{P}^i|} \lambda^i_k z^{i,k}_{[m]} + \sum_{r=1}^{q_i} u^i_r \right) \leq b \tag{27b}$$

$$\sum_{k=1}^{|\mathcal{P}^i|} \lambda^i_k + \sum_{r=1}^{q_i} \mu^i_r = 1, \qquad i \in \mathcal{M} \tag{27c}$$

$$w^i_r - \hat{\gamma}^T u^i_r \geq (\tilde{r}_i + \hat{\alpha}_i)\mu^i_r, \qquad i \in \mathcal{M}, \ r = 1, \dots, q_i \tag{27d}$$

$$\lambda \geq \mathbf{0}, \ \mu \geq \mathbf{0}, \ w \text{ free}, \ u \text{ free}. \tag{27e}$$

Using dual variables $\gamma \leq \mathbf{0}$ to denote the LP dual multipliers for constraint (27b), α (unrestricted in sign) to denote the vector of LP dual multipliers for the constraint (27c), and $\eta^i_r \geq 0$ to denote the LP dual multipliers for each pricing cut in (27d) indexed by $i \in \mathcal{M}$, $r = 1, \dots, q_i$, we formulate the LP dual to the above LP:

$$\max \ b^T \gamma + \mathbf{1}^T \alpha_i \tag{28a}$$

$$\text{s.t. } \gamma^T z^{i,k}_{[m]} + \alpha_i \le z^{i,k}_{m+1}, \qquad i \in \mathcal{M},\ k = 1,\dots,|\mathcal{P}^i| \tag{28b}$$
$$\eta \le \mathbf{1} \tag{28c}$$
$$\gamma - \hat{\gamma} \le \mathbf{0} \tag{28d}$$
$$\alpha_i - (\tilde{r}_i + \hat{\alpha}_i)\eta^i_r \le 0 \qquad i \in \mathcal{M},\ r = 1,\dots,q_i \tag{28e}$$
$$\gamma \le \mathbf{0},\ \alpha \text{ free},\ \eta \ge \mathbf{0}. \tag{28f}$$

Setting $\gamma := \hat{\gamma}$, $\alpha := \hat{\alpha}$, $\eta := \mathbf{1}$ yields a feasible solution to the above LP: (28b) follows from (24) and since $z^{i,k} \in \mathcal{Z}^i$ for all i, k; (28c) and (28d) follow immediately from the definitions of η and γ; and (28e) follows from the definitions and from (25). By (26) and since $\gamma := \hat{\gamma}$, $\alpha := \hat{\alpha}$, $\eta := \mathbf{1}$ yields a feasible solution to the above LP, which is dual to the relaxation of the LP relaxation of (LB-M), its objective value, which is $\hat{\gamma}^T b + \hat{\alpha}^T \mathbf{1}$, satisfies

$$\nu^{IP}_{LP} = \hat{\gamma}^T b + \hat{\alpha}^T \mathbf{1} \le \nu^{LB}_{R} \le \nu^{LB}_{LP}$$

and the result follows. □

B Initial region creation example

As an example of Algorithm 2, consider a case with three objectives where two NDPs, namely $(4,4,4)$ and $(1,8,1)$, are discovered for MOP(i), which has supernal point $(50,50,50)$. The regions created during Algorithm 2 (where $\epsilon = 1$) are shown in Figure 6 in a tree. A region is represented by a triplet in brackets, where each component represents an upper bound on the corresponding objective, e.g., $\langle \delta_1, \delta_2, \delta_3 \rangle$ defines the polyhedral region $\{z \in \mathbb{R}^3 : z_j \le \delta_j, j = 1,2,3\ \}$. Level ℓ of the tree corresponds to the ℓ^{th} iteration of the algorithm and therefore to the list of regions in that iteration. The boxed regions are the ones returned at the end of the fifth step of the algorithm. The shaded boxes represent redundant regions, which are contained in other regions, and can safely be removed in the last step of the algorithm.

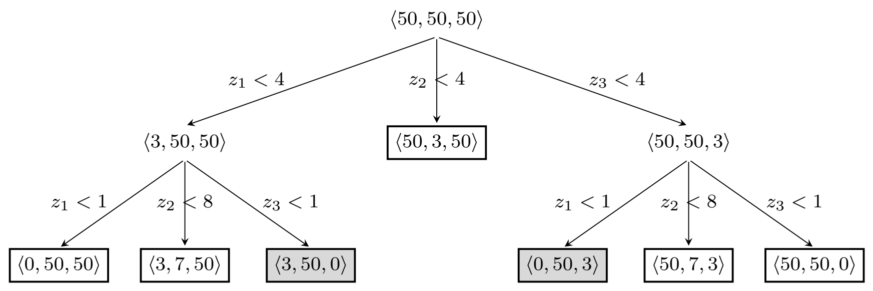


Fig. 6 Algorithm 2 applied to MOP(i) with supernal point $(50,50,50)$, provided $\epsilon = 1$ and $\mathcal{P}^i = \{(4,4,4),(1,8,1)\}$

We note that the decomposition technique used in Algorithm 2 is the same as the so-called *full m-split* in [14] when all the initial lower and upper bounds are taken as negative infinity and the supernal point, respectively. In [14], the authors only generate hypercubes as the regions, which are called *boxes*. They point out that the decomposition algorithm based on full m-split typically creates nested, and hence, *redundant*, boxes. The authors analyze conditions under which redundant boxes occur, and suggest to detect and remove them from the decomposition immediately in every iteration. In the case illustrated in Figure 6, as $(3,50,0) \le (50,50,0)$, the region represented by $\langle 3,50,0 \rangle$ is redundant, as is the region represented by $\langle 0, 50, 3\rangle$, since $(0, 50, 3) \leq (0, 50, 50)$. Note that the tree node corresponding to the region represented by $\langle 50, 3, 50\rangle$ has no descendants, since $(1, 8, 1) \not\leq (50, 3, 50)$.

We also note that, in the case of a single coupling constraint, (i.e., $m = 1$), redundancy does not occur, as any NDP belongs to exactly one of the regions on the list at any point in Algorithm 2.

C Examples for branch-and-price comparison

C.1 Column Generation

Example 1 Consider the following integer program with two identical blocks and a single linking constraint:

$$\min\{x_1^1 + x_1^2 : x^1 \in \mathcal{X}^1, x^2 \in \mathcal{X}^2, x_2^1 + x_2^2 \leq 3\}, \text{ where}$$

$$\mathcal{X}^1 = \mathcal{X}^2 = \mathcal{X} := \{x \in \mathbb{Z}^2 : 2x_1 + 3x_2 \geq 6, 0 \leq x_1, x_2 \leq 3\}.$$

For any block $i \in \{1, 2\}$, the set of NDPs for MOP(i) is $\mathcal{N}^i = \{(0, 3), (1, 2), (2, 0)\}$, while the set of extreme points of the convex hull of its feasible set is $\mathcal{E}^i = \{(0, 2), (3, 0), (0, 3), (3, 3)\}$. The IP has optimal value 2, with two, symmetric, optimal solutions: $x^1 = (2, 1)$, $x^2 = (0, 2)$ and $x^1 = (0, 2)$, $x^2 = (2, 1)$. Importantly, $(2, 1) \in \mathcal{X}$ maps to the NDP $(1, 2)$, and is required in an optimal solution, but $(2, 1)$ is interior to $\operatorname{conv}(\mathcal{X})$.

The MOP-based reformulation of this IP, (7), is

$$\min\{ \sum_{i=1,2} (2\lambda_2^i + 3\lambda_3^i) \ : \ \sum_{i=1,2} (2\lambda_1^i + \lambda_2^i) \leq 3, \ \lambda_1^i + \lambda_2^i + \lambda_3^i = 1, \ \lambda^i \in \{0,1\}^3, \ i = 1, 2\},$$

which has two alternative optimal solutions: $\lambda^1 = (0, 1, 0)$, $\lambda^2 = (1, 0, 0)$ and $\lambda^1 = (1, 0, 0)$, $\lambda^2 = (0, 1, 0)$, corresponding to the two optimal solutions to the original IP, respectively.

By contrast, the Dantzig-Wolfe reformulation LP master problem is

$$\min\{ \sum_{i=1,2} (3\lambda_2^i + 3\lambda_4^i) \ : \ \sum_{i=1,2} (2\lambda_1^i + 3\lambda_3^i + 3\lambda_4^i) \leq 3, \ \lambda_1^i + \lambda_2^i + \lambda_3^i + \lambda_4^i = 1, \ \lambda^i \geq \mathbf{0}, \ i = 1, 2\}.$$

Treating this as an IP, by requiring $\lambda^i \in \{0, 1\}^4$ for $i = 1, 2$, does not give a valid reformulation of the original IP. It has optimal value 3, given by, for example, $\lambda^1 = (1, 0, 0, 0)$ and $\lambda^2 = (0, 1, 0, 0)$. Instead, solving the LP master problem gives optimal value 1.5, with (one possible) solution $\lambda^1 = (1, 0, 0, 0)$ and $\lambda^2 = (0.5, 0.5, 0, 0)$, corresponding to $x^1 = (0, 2)$, $x^2 = (1.5, 1)$. Branching on x_1^2 gives two new IPs: on one branch, $x_1^2 \leq 1$ is applied, while on the other, $x_1^2 \geq 2$ is applied. On this latter branch, $\mathcal{E}^2 = \{(2, 1), (3, 0), (2, 3), (3, 3)\}$, and the solution $x^2 = (2, 1)$ is now "exposed": the optimal solution to the Dantzig-Wolfe LP master problem at this node of the branch-and-price tree is that corresponding to $x^1 = (0, 2)$, $x^2 = (2, 1)$, an optimal solution to the original IP. However the former branch, with $\mathcal{E}^2 = \{(0, 2), (1, 2), (0, 3), (1, 3)\}$, returns LP master solution corresponding to $x^1 = (1.5, 1)$, $x^2 = (0, 2)$, the LP solution symmetric to that at the root node. Thus this branch needs to be explored further before optimality can be proved.

Example 2 Consider the following integer program with M identical blocks, each corresponding to an assignment problem, and a single linking constraint:

$$\min\Big\{ \sum_{i\in\mathcal{M}} x_{11}^i : \ x^i \in \mathcal{X}_{AP}, \, i \in \mathcal{M} \text{ and } \sum_{i\in\mathcal{M}} x_{22}^i \leq M \Big\}, \text{ where}$$

$$\mathcal{X}_{AP} = \Big\{ x \in \{0, 1\}^{n\times n} : \sum_{h=1}^{n} x_{h\ell} = 1, \ell = 1, \dots, n, \ \sum_{\ell=1}^{n} x_{h\ell} = 1, h = 1, \dots, n \Big\}.$$

Note that the linking constraint is indeed redundant as x variables are all binary. For $i \in \mathcal{M}$, we have

$$\mathrm{MOP}(i) : \min\{x_{22}, x_{11}\} \text{ s.t. } x \in \mathcal{X}_{AP} \;\Rightarrow\; \mathcal{N}^i = \{(0,0)\},$$

thus (IP-M) consists of only M columns. On the other hand, the extreme points of the assignment polytope (also known as the Birkhoff polytope) correspond to 0-1 matrices of size $n \times n$ with exactly one 1 on every row and every column. Under the objective of $\mathrm{MOP}(i)$ subproblem, these $n!$ extreme points yield 2^2 different images (all binary permutations of size two), which means that the Dantzig-Wolfe reformulation of the problem has $4M$ columns in total. Therefore, the difference between the number of columns in two formulations becomes larger as the number of blocks increases. Lastly, we note that the difference can be made even larger by adding more linking constraints. □

Example 3 Consider the following integer program with two identical blocks and a single linking constraint:

$$\min\{x^1_2 + x^2_2 : x^1 \in \mathcal{X}, x^2 \in \mathcal{X}, x^1_1 + x^2_1 \le 3\}, \text{ where}$$

$$\mathcal{X} = \{x \in \mathbb{Z}^2 : x_1 + 2x_2 \ge 7,\ 0 \le x_1, x_2 \le 4\}.$$

For any block $i \in \{1,2\}$, we have $\mathcal{N}^i = \{(0,4),(1,3),(3,2)\}$. Suppose that we initialize the LP relaxation (in Step 1) with dummy columns corresponding to ideal point $z^I = (0,2)$ with sufficiently big objective coefficients, e.g., 100. Then, at the very first iteration of the standard column generation, as the optimal dual multiplier of the linking constraint happens to be zero, both $(3,2)$ and $(4,2)$ are the columns with the most negative reduced cost. Therefore, we might be adding the column $(4,2)$ which is weakly nondominated. □

C.2 Branching

Example 4 We consider a problem, with a single linking constraint, of the form

$$\min\Big\{\sum_{i=1}^{2}(c^i)^\top x^i :\ x^i \in \mathcal{X}_{AP},\, i = 1,2 \text{ and } \sum_{i=1}^{2} A^i x^i \le b\Big\},$$

where

$$\mathcal{X}_{AP} = \Big\{x \in \{0,1\}^{n\times n} : \sum_{h=1}^{n} x_{h\ell} = 1, \ell = 1,\dots,n,\ \sum_{\ell=1}^{n} x_{h\ell} = 1, h = 1,\dots,n\Big\}.$$

We generate an instance with $n = 10$ by letting the entries of c^i and A^i for $i = 1,2$ to be integers drawn uniformly from $\{0,\dots,40\}$. We set $b = 200$. The data for the resulting instance is given below, with matrix data given with rows concatenated. So, for example, $c^1_{3,7} = 6$ is given by the 27th entry in the c^1 vector.
A^1 = [3, 3, 4, 19, 1, 21, 10, 36, 17, 32, 34, 36, 3, 19, 8, 34, 30, 0, 5, 23, 19, 5, 32, 40, 2, 13, 4, 20, 31, 12, 15, 36, 17, 19, 14, 19, 40, 24, 14, 19, 17, 9, 14, 23, 28, 25, 16, 17, 27, 23, 1, 5, 28, 33, 5, 33, 8, 11, 12, 0, 23, 27, 36, 0, 5, 9, 21, 7, 35, 37, 26, 13, 6, 1, 36, 34, 26, 14, 11, 12, 37, 14, 20, 27, 9, 25, 19, 17, 38, 33, 17, 20, 20, 12, 22, 27, 23, 4, 36, 17]
c^1 = [33, 20, 25, 23, 8, 13, 32, 4, 14, 13, 8, 11, 11, 22, 33, 5, 23, 18, 19, 24, 21, 40, 0, 28, 4, 23, 6, 4, 2, 4, 20, 37, 26, 4, 20, 34, 19, 13, 38, 33, 26, 5, 3, 37, 30, 38, 3, 12, 15, 22, 38, 37, 23, 39, 26, 28, 23, 32, 34, 25, 36, 13, 22, 21, 17, 3, 16, 38, 16, 13, 30, 3, 21, 35, 1, 10, 32, 4, 24, 8, 29, 23, 6, 11, 23, 33, 39, 5, 24, 34, 31, 22, 6, 14, 4, 25, 17, 21, 22, 33]
A^2 = [27, 12, 32, 32, 21, 8, 23, 29, 22, 7, 12, 3, 40, 39, 19, 28, 33, 18, 17, 36, 18, 4, 21, 5, 29, 29, 9, 20, 0, 31, 10, 27, 2, 3, 18, 25, 14, 3, 13, 36, 12, 28, 0, 14, 26, 19, 3, 18, 40, 22, 15, 17, 27, 39, 23, 17, 27, 34, 38, 27, 26, 9, 13, 31, 14, 33, 15, 30, 38, 31, 25, 12, 18, 28, 26, 5, 8, 31, 25, 7, 13, 2, 27, 40, 0, 9, 18, 29, 4, 17, 15, 31, 28, 30, 23, 2, 24, 38, 34, 22]
c^2 = [30, 8, 17, 2, 11, 2, 30, 27, 30, 11, 9, 23, 6, 37, 9, 0, 15, 28, 39, 12, 30, 18, 10, 24, 7, 39, 21, 40, 29, 31, 16, 21, 0, 35, 23, 11, 37, 12, 40, 28, 25, 8, 10, 31, 7, 21, 31, 22, 9, 31, 36, 0,

10, 8, 24, 19, 6, 4, 20, 38, 38, 37, 18, 38, 33, 0, 9, 32, 14, 10, 19, 0, 21, 32, 31, 28, 12, 23, 11, 21, 15, 7, 22, 27, 15, 5, 5, 23, 12, 25, 20, 11, 23, 38, 8, 16, 40, 19, 7, 15]
$b = 200$

For this instance, the NDFs of MOP subproblems are given in Figure 7. The circle points in the figure correspond to the columns generated at Step 1 of Algorithm 1, i.e., when the LP relaxation of (IP-M) is solved. We note that these columns are also the images of the columns generated when the LP relaxation of the Dantzig-Wolfe reformulation is solved via column generation, since the initial steps of the classical branch-and-price and our MOP-based branch-and-price coincide. Lastly, the circle points with a cross in the figure represent the NDPs whose corresponding columns have a positive weight in the optimal LP solution. Only the LP relaxation for block $i = 2$ is fractional: there is positive weight on the NDPs

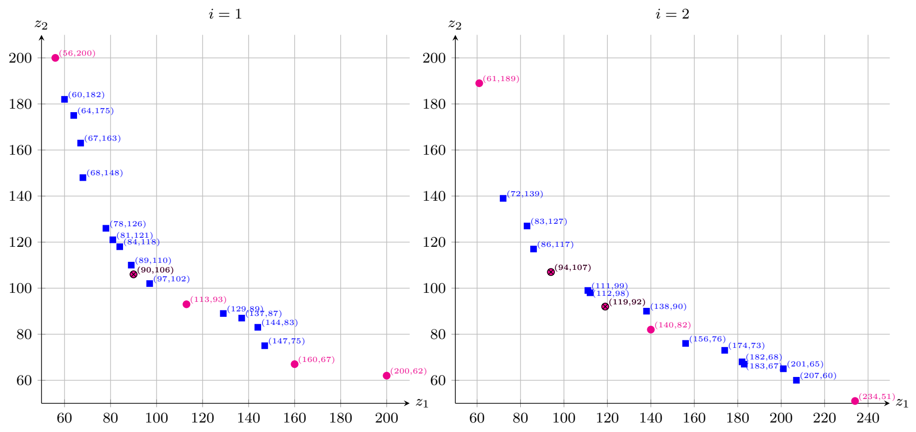


Fig. 7 NDF of MOP(i) subproblems which are biobjective assignment problems ($\mathcal{N}^1$ on the left, $\mathcal{N}^2$ on the right).

$(94, 107)$ and $(119, 92)$, while for block $i = 1$, the NDP $(90, 106)$ is selected.

For this instance, the (not quite complete) branch-and-bound tree obtained by using the traditional column generation branching and the branch-and-bound tree obtained using the new *NDP-based* branching rule are provided below. In both cases, we use the best bound search strategy, since this is guaranteed to give the smallest tree.

Traditional column generation branching would branch on a fractional $x^i_{h\ell}$ variable. On one branch, all columns corresponding to solutions with $x^i_{h\ell} = 1$ are removed and $x^i_{h\ell}$ is fixed to 0 fixed in the column generation subproblem for i, which does not change its assignment problem structure. On the other branch, all columns corresponding to solutions with $x^i_{h\ell} = 0$ are removed and $x^i_{h\ell}$ is fixed to 1 fixed in the column generation subproblem, which again preserves its assignment problem structure. There could very well be many possible i, h, ℓ indices, and there seems to be little to guide which to choose. Also, changing just one variable in the assignment subproblems may not have a big effect, if there are many alternative assignment solutions with fairly similar resource consumption and cost values, which can occur if the problem exhibits symmetry or near-symmetry. Solving this instance using the standard approach of selecting the branching variable $x^i_{h\ell}$ to be that which is most fractional in the LP solution, breaking ties by choosing the variable with the smallest index in terms of the lexicographical order of i, h, ℓ, we obtain the incomplete branch-and-bound tree given in Figure 8. Note the tree has 103 nodes, with an optimal solution first found at depth 11, but has one node that still needs to be explored.

Several branches in this tree have subproblem feasible regions that contain no NDPs. For instance, such a case occurs at a node at depth three, after fixing $x^2_{1,6} = 0, x^2_{6,2} = 1, x^2_{2,1} = 0$, in that order. These branching constraint restrict the feasible set of MOP(2) to a region

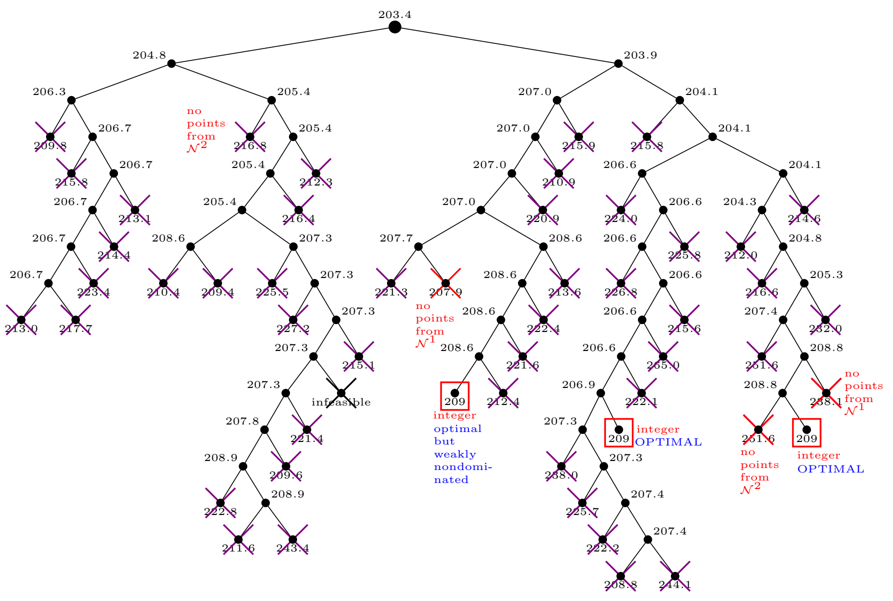


Fig. 8 The branch-and-bound tree obtained by using the traditional column generation branching rule and the best-first search strategy. Left (right) branch corresponds to fixing a decision variable to zero (one)

containing no NDPs. However, this branch has master LP objective value 216.8, and so is pruned once the optimal solution is discovered, having value 209. Another example is seen at depth seven, at the node obtained after fixing $x^2_{1,6} = 1, x^1_{3,7} = 0, x^1_{1,2} = 0, x^1_{1,1} = 0, x^1_{1,3} = 0, x^1_{1,5} = 0, x^1_{4,1} = 1$ in that order. Here, the feasible set of MOP(1) contains no NDPs. This node has LP objective value 207.9, and so needs to be explored before optimality is confirmed.

By contrast, consider NDP-based branching. Let λ^i_k be fractional in the optimal LP solution. Then, there must exist $k' \neq k$ with $\lambda^i_{k'}$ also fractional and at least one index j for which $z^{i,k}_j \neq z^{i,k'}_j$. Here we preference the choice of $j = m + 1 = 2$, which denotes the objective component, since it is most likely to increase the lower bound quickly. In this case, the NDP-based branch rule produces two branches (as $m = 1$) via the disjunction

$$A^i x^i \leq z^{i,k}_1 - \epsilon \quad \bigvee \quad (c^i)^\top x^i \leq z^{i,k}_2$$

where $\epsilon = 1$ can be used for pure IPs and k is the index of the fractionally selected NDP for MOP(i) having smallest objective value. For example, at the root node of the instance, $i = 2$ is chosen, as both $(94, 107)$ and $(119, 92)$ are selected fractionally in the LP relaxation. Since the latter has smaller objective value, it is used to create the branching disjunction, which is thus

$$A^2 x^2 \leq 118 \quad \bigvee \quad (c^2)^\top x^2 \leq 92.$$

This branching rule is implemented by adding one of the branching constraints to MOP(i), removing the columns violating this constraint from the master LP, and re-solving the master LP to optimality, where more NDPs are possibly generated.

The full branch-and-bound tree obtained for this instance, given in Figure 9, has only 11 nodes in total, four of which are pruned by infeasibility. The optimal solution is found at depth four. In Figure 9, boxes with solid borders represent integral LP solutions, whereas boxes with dashed borders represent fractional LP solutions, where the corresponding NDPs for $i = 1$ and $i = 2$ are given in the left and right columns, respectively, and the optimal value is denoted by "obj". The optimal solution appears in the shaded box, at depth four. Constraints added to the MOP subproblems are given on the arrows, where $z_1^i := A^i x$ and $z_2^i := (c^i)^\top x$ are used for convenience.

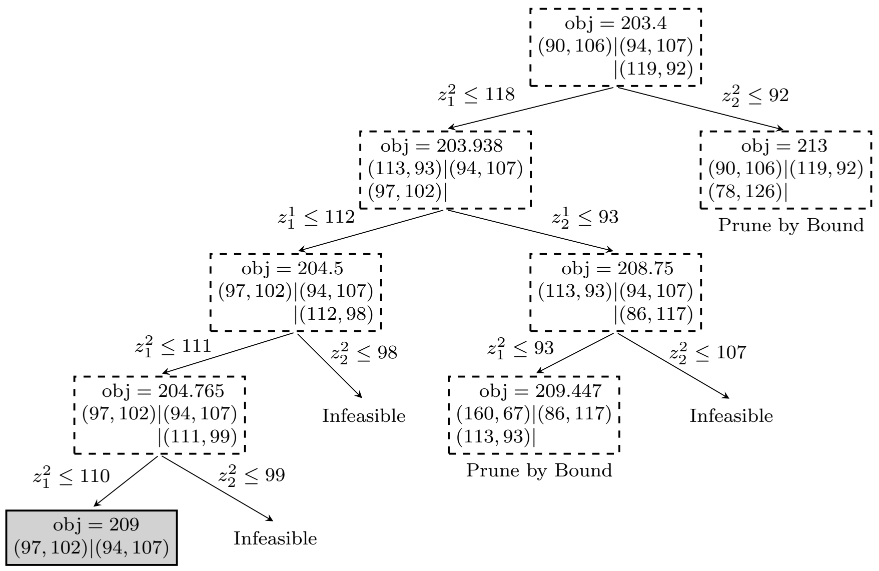


Fig. 9 The branch-and-bound tree obtained by using the new *NDP-based* branching rule and the best-first (also known as best bound) tree search strategy.

D Computational Results

D.1 Redundancy check

We examine the effect of detecting and removing redundant regions. Specifically, for KC and MMKP instances, we turn off the redundancy check in the algorithm setting found to be best overall: no pricing cuts, $\beta = \mathbf{1}$ throughout and the Conservative region refinement strategy. In Table 3, the total CPU time (in seconds) and the number (LB-M) iterations to solve an instance are given in the columns labeled as "Time" and "It", respectively. The (geometric for KC, arithmetic for MMKP) average, over M blocks, of the initial number of regions, i.e., the number of regions obtained at the end of Step 2 of Algorithm 1, and of the total number of regions generated throughout the algorithm are provided in the columns labeled as "Init" and "Tot", respectively. Lastly, the (geometric for KC, arithmetic for MMKP) average number of redundant regions detected and removed at Step 2 and Step 3 are given in the last two columns labeled as "St2" and "St3", respectively.

	Without redundancy check				With redundancy check					
Instance	Time	It	Init	Tot	Time	It	Init	Tot	St2	St3
3_850,950	49	14	9.5	613.1	14	14	6.3	179.3	3.2	71.2
3_900,950	51	16	9.8	649.6	14	16	6.3	178.9	2.5	71.4
3_950,950	51	16	9.8	649.6	14	16	6.3	178.9	2.5	71.4
4_1200,1200	48	16	8.5	386.4	16	16	5.9	145.1	0.0	53.6
4_1300,1300	51	16	8.3	402.9	15	16	5.9	146.6	0.0	50.3
4_1400,1400	55	16	6.6	499.2	16	16	5.1	149.0	0.0	54.8
5_1500,1600	59	16	7.9	414.1	21	16	5.6	155.4	2.3	57.4
5_1600,1500	64	16	7.2	448.2	20	16	5.3	156.1	0.0	58.3
5_1600,1550	59	16	7.8	414.5	20	16	5.7	153.4	0.0	58.5
6_1820,1900	71	16	6.7	429.1	24	16	5.3	153.7	0.0	57.9
6_1900,1900	62	16	7.3	368.4	22	16	5.6	153.2	0.0	55.0
6_1900,1850	70	16	6.7	432.6	23	16	5.3	153.9	0.0	58.2
7_2250,2250	76	16	7.3	411.1	28	16	5.4	161.9	0.0	57.9
8_2600,2500	94	16	7.5	437.2	30	16	5.4	150.7	0.0	57.2
9_2800,2900	97	16	6.3	428.7	34	16	5.0	155.5	0.0	58.6
10_3100,3150	104	16	6.3	442.0	39	16	5.0	155.9	0.0	60.2
INST01	4.1	9	9.2	185.7	2.1	9	6.5	51.7	2.6	29.3
INST02	4.5	9	5.3	219.8	2.1	9	4.8	57.3	0.5	32.8
INST03	4.9	9	9.9	155.5	2.6	9	7.0	48.7	2.7	27.7
INST04	6.4	9	8.8	190.3	3.3	9	6.4	54.6	2.3	32.1
INST05	6.2	9	8.0	193.1	3.2	9	6.1	52.8	1.8	29.9
INST06	6.4	9	8.0	192.1	3.2	9	6.1	52.6	1.9	29.8
INST07	7.0	9	8.7	179.9	3.7	9	6.4	50.7	2.3	29.2
INST08	7.0	9	8.3	194.7	3.6	9	6.2	51.9	2.1	30.5
INST09	6.9	9	8.4	195.5	3.5	9	6.3	52.1	2.1	30.5
INST10	7.7	9	8.4	189.8	3.8	9	6.3	52.3	2.1	30.1
INST11	7.3	9	8.4	192.0	3.8	9	6.3	52.6	2.1	30.3
INST12	8.1	9	8.7	178.5	4.2	9	6.5	52.5	2.1	30.5

Table 3 Redundancy check results for KC instances, and small MMKP instances with $|\mathcal{K}| = 3$ (Setting: No pricing cuts, $\beta = \mathbf{1}$, conservative region refinement)

For KC, we observe that when redundant regions are removed, (LB-M) is initialized with almost half as many regions, the total number of generated regions is reduced by an order of magnitude, and accordingly the algorithm converges in a significantly smaller number of iterations, and so is much faster. Moreover, the average time spent on detecting redundant regions is only 0.002 seconds per instance. Lastly, we note that the number of initial regions in the case with redundancy check plus the number of redundant regions removed at Step

2 is not necessarily equal to the number of initial regions in the case without redundancy check, as redundant regions are removed at the end of each refinement step.

For the easiest MMKP instances, although the solution times are still small in the absence of the redundancy check, they are roughly doubled. The redundancy check times are again negligible, constituting almost always less than 1%, and at most 1.25%, of the solution times for those small instances. More importantly, without detecting and removing redundant regions, none of the large MMKP instances, INST13–INST20, can be solved in the time limit of an hour.

We also illustrate the impact of the redundancy check in a different algorithmic setting in Table 4 for the KC instances. In this version of the algorithm, where we apply both the pricing cuts and the aggressive region refinement, we observe that when redundant regions are removed, (LB-M) is initialized with almost half as many regions, the total number of generated regions is reduced by an order of magnitude, and accordingly the algorithm converges in a significantly smaller number of iterations, and so is much faster. Moreover, the average time spent on detecting redundant regions is only 0.0003 seconds per instance.

	Without redundancy check				With redundancy check					
Instance	Time	It	Init	Tot	Time	It	Init	Tot	St2	St3
3_850,950	139	68	19.9	317.9	5	19	9.6	33.1	5.8	38.0
3_900,950	29	35	15.3	223.0	2	13	8.3	23.9	4.6	25.1
3_950,950	163	50	38.6	484.9	5	16	12.3	38.7	8.4	47.4
4_1200,1200	408	67	23.6	340.7	8	29	9.9	37.9	7.7	48.3
4_1300,1300	249	41	27.9	199.5	5	19	10.5	20.2	9.9	31.2
4_1400,1400	67	52	26.0	126.1	2	14	10.8	11.4	8.9	22.2
5_1500,1600	95	54	12.4	295.4	8	24	7.7	42.1	3.3	45.0
5_1600,1500	476	82	16.5	384.1	12	33	8.5	47.3	5.4	50.4
5_1600,1550	351	55	21.8	317.9	8	22	9.4	39.7	7.5	49.1
6_1820,1900	1424	116	14.4	380.5	53	50	8.2	51.8	3.8	48.2
6_1900,1900	745	87	28.1	272.7	13	32	10.9	28.7	8.6	31.1
6_1900,1850	107	61	17.1	403.6	11	31	8.6	47.9	5.9	46.7
7_2250,2250	2057	142	27.9	314.6	23	45	10.9	34.8	8.5	36.3
8_2600,2500	1474	118	14.9	289.2	31	39	8.1	31.9	4.7	32.0
9_2800,2900	2182	128	13.0	390.9	98	70	7.5	55.3	3.6	54.1
10_3100,3150	806	117	12.1	328.7	66	44	7.5	50.6	3.1	47.6

Table 4 Additional redundancy check results for KC instances (Setting: Pricing cuts, β guided by (LB-M) while switching to the vector of ones as needed, aggressive region refinement)

In summary, we observe that redundancy check times are negligible, and that when redundant regions are removed, (LB-M) is initialized with a significantly smaller number of regions, the total number of generated regions is reduced by an order of magnitude, and accordingly the algorithm converges much faster. Without the redundancy check, none of the large MMKP instances can be solved in the time limit. Lastly, the impact of redundancy check is magnified if a more aggressive region refinement strategy is used. We conclude that removing redundant regions plays a crucial role in attaining efficiency.

D.2 Algorithmic choices

Solution times are reported in seconds (or "TL" standing for time limit). Best times are indicated in bold font.

	$\beta = \mathbf{1}$				Based on duals from (LB-M)											
	Pricing Cuts		No Cuts		Pricing Cuts						No Cuts					
					All One		Small Pos		Leave As Is		All One		Small Pos		Leave As Is	
Instance	Agg	Con	Agg	Con	Agg	Con	Agg	Con	Agg	Con	Agg	Con	Agg	Con	Agg	Con
3_850,950	5.8	4.4	11.4	13.7	4.8	**3.2**	4.8	3.7	4.8	3.9	11.3	14.5	11.9	15.8	11.8	16.3
3_900,950	2.6	1.6	11.0	14.4	2.3	**1.3**	2.2	1.4	2.2	1.5	11.1	13.9	11.6	15.6	11.4	16.0
3_950,950	7.7	2.2	11.8	14.3	5.3	**1.7**	5.3	1.8	5.3	1.9	11.6	14.8	11.5	16.8	11.3	16.0
4_1200,1200	13.5	8.2	12.8	16.0	8.2	**3.5**	7.8	3.9	7.8	3.7	12.8	16.2	12.5	17.8	12.7	18.1
4_1300,1300	4.4	2.8	12.9	15.2	5.2	**1.5**	5.3	**1.5**	5.2	**1.5**	13.4	15.9	13.8	17.6	13.5	17.1
4_1400,1400	2.8	2.4	12.3	16.4	2.4	**1.5**	2.6	2.1	2.5	2.0	12.4	16.3	12.9	19.5	13.2	18.4
5_1500,1600	14.0	13.5	15.3	20.8	7.9	3.9	7.9	**3.8**	8.0	**3.8**	15.7	20.1	15.9	22.4	15.7	22.7
5_1600,1500	18.4	44.9	15.2	19.6	12.1	8.2	12.2	**7.4**	12.4	7.7	15.7	20.3	15.7	23.8	15.4	24.1
5_1600,1550	9.1	7.2	15.7	20.1	8.0	**5.3**	8.2	5.4	8.1	5.5	14.7	19.7	15.8	23.3	15.2	22.6
6_1820,1900	43.6	75.8	17.3	23.7	53.3	65.8	52.6	72.9	52.3	67.7	**17.3**	21.7	18.1	25.4	17.8	24.6
6_1900,1900	18.1	10.4	17.3	22.3	12.5	5.5	12.5	5.4	12.6	**5.4**	16.7	21.6	17.5	24.7	17.2	24.2
6_1900,1850	30.5	35.1	17.7	23.1	11.5	16.3	**11.1**	12.6	11.2	13.0	16.3	21.4	16.5	25.0	16.3	24.6
7_2250,2250	40.5	30.5	22.4	27.8	23.4	7.9	23.7	**7.8**	25.9	**7.8**	20.5	26.3	21.0	29.5	20.6	28.9
8_2600,2500	62.3	130.2	22.2	30.3	30.9	35.7	28.7	34.6	28.1	36.8	**21.1**	28.5	21.4	32.2	21.4	31.8
9_2800,2900	36.7	67.7	25.9	34.5	97.6	123.6	95.9	103.8	99.6	97.1	**26.5**	31.9	27.6	36.2	27.3	35.6
10_3100,3150	56.6	222.7	29.0	38.8	66.5	105.5	74.8	106.2	69.0	91.5	**26.5**	35.3	27.0	39.5	26.8	38.9
INST01	445.7	960.7	**1.4**	2.1	329.2	298.9	379.8	289.8	383.6	287.6	1.6	2.3	1.5	2.2	1.5	2.3
INST02	17.3	9.3	**1.4**	2.1	21.3	9.9	16.6	15.0	9.8	8.7	1.6	2.3	1.5	2.3	1.5	2.3
INST03	161.7	1099.5	**1.8**	2.6	78.1	112.6	88.0	101.2	91.7	108.7	1.9	2.8	**1.8**	2.8	1.9	2.9
INST04	911.2	1484.6	**2.1**	3.3	774.6	568.1	790.8	545.3	760.5	517.5	2.4	3.6	2.4	3.4	2.4	3.3
INST05	2302.2	TL	**2.0**	3.2	2080.1	1265.0	2064.1	1203.1	2019.2	1206.2	2.2	3.7	2.1	3.5	2.1	3.4
INST06	259.4	314.3	**2.3**	3.2	296.1	161.6	454.5	101.8	323.3	95.3	2.4	3.4	2.5	3.4	2.4	3.4
INST07	2649.0	3526.1	**2.7**	3.7	2391.9	468.8	1857.9	515.4	2079.7	487.0	**2.7**	3.9	2.8	3.8	**2.7**	3.8
INST08	1203.0	1039.8	2.8	3.6	1249.9	341.6	1332.6	355.4	1315.4	303.6	2.8	3.7	2.6	3.7	**2.5**	3.6
INST09	585.6	520.2	2.7	3.5	580.5	265.4	670.9	517.6	685.4	252.6	**2.6**	3.7	**2.6**	3.7	2.7	3.7
INST10	689.7	994.4	**2.8**	3.8	510.1	132.6	612.3	108.2	664.5	130.6	3.0	4.0	3.0	4.0	**2.8**	4.0
INST11	1388.5	2174.7	**2.7**	3.8	1352.7	449.5	1545.2	456.7	1425.9	371.9	2.8	4.1	2.8	4.1	2.8	4.0
INST12	296.7	2017.3	**2.8**	4.2	376.3	141.9	295.4	135.5	285.9	157.6	3.0	4.4	3.2	4.5	3.2	4.5

Table 5 Time comparison of alternative settings for KC instances, and small MMKP instances with $|\mathcal{K}| = 3$

Results for small MMKP instances with more linking constraints, without the use of pricing cuts, are provided in Table 6, where the best CPU times are displayed in bold. While the β selection strategy comparison remains the same as in Table 5, the region refinement domination result is reversed: the Conservative strategy reduces solution times by 32% on the average. More specifically, we find that in MMKP instances, the duals obtained from (LB-M) always include a zero element, and the alternatives for β that we consider in this case almost always lead to the discovery of the same NDPs, and thus do not change the course of the algorithm. On the other hand, the Conservative refinement strategy doubles the number of iterations but yields time savings.

			$\beta = \mathbf{1}$		Based on duals from (LB-M)					
					All One		Small Pos		Leave As Is	
Instance	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{J}\rvert$	Agg	Con	Agg	Con	Agg	Con	Agg	Con
INST01	50	10	**12.2**	13.8	13.7	17.1	13.7	13.8	13.4	14.6
INST02	50	10	17.4	**8.4**	19.1	10.7	19.1	9.9	18.0	10.0
INST03	60	10	19.5	**13.3**	20.9	16.9	20.5	14.6	18.9	14.7
INST04	70	10	32.6	**17.6**	35.0	21.7	37.2	21.2	32.5	23.1
INST05	75	10	34.5	**20.1**	35.2	22.8	34.2	25.0	32.1	25.2
INST06	75	10	29.4	22.0	31.3	23.7	33.3	**20.9**	30.4	24.0
INST07	80	10	27.7	**21.9**	29.7	25.0	30.3	24.0	30.1	26.7
INST08	80	10	35.9	**22.0**	40.6	26.6	38.2	25.9	37.2	25.4
INST09	80	10	31.6	**18.5**	38.8	22.4	31.9	23.3	32.7	22.2
INST10	90	10	36.0	**19.1**	38.5	23.6	34.1	22.4	33.2	29.2
INST11	90	10	35.9	**17.2**	40.9	20.5	37.8	20.6	38.6	21.4
INST12	100	10	46.5	**18.8**	51.7	23.8	48.5	24.8	46.5	24.2

Table 6 Time comparison of alternative settings for small MMKP instances with $\lvert\mathcal{K}\rvert = 5$ (Setting: No pricing cuts)

D.3 $\boldsymbol{\beta}$ vector selection

In Table 7, we provide additional, more detailed, results for the KC instances, to provide insights into the trade-offs in play for alternative choices of β in the setting with pricing cuts and the Aggressive region refinement strategy. In addition to the solution times ("Time" columns) and the number of (LB-M) iterations ("It" columns), we provide the average number of times Line 16 is executed in Algorithm 3 (St16 columns) and the number of iterations performed in Algorithm 3 ("It2" columns) *per (LB-M) iteration*. Also, for the fallback option when β is supplied by the (LB-M)duals, we give the number of times we switch to $\beta = \mathbf{1}$ in the last column of the table.

From these results, we see that the algorithm gets into Line 16 less frequently, which in turn yields reductions in the number of (LB-M) iterations, in most instances, and in the number of iterations performed in Algorithm 3. However, these do not always translate into reductions in overall solution times, which are still lower for the hardest instances when using the constant $\beta = \mathbf{1}$ option.

D.4 Aggressive vs conservative refinement

In Table 8, we provide total solution time (Time), the number (LB-M) iterations (It); and the (geometric) average, over all blocks, of the number of regions obtained at the end of Step 2 (Init), and of the total number of regions generated throughout the algorithm (Tot).

	$\beta = \mathbf{1}$				Based on duals from (LB-M) & All One				
Instance	Time	It	St16	It2	Time	It	St16	It2	Switch
3_850,950	5.8	18	0.00	6.5	4.8	19	0.00	6.0	5
3_900,950	2.6	14	0.00	5.3	2.3	13	0.00	4.2	5
3_950,950	7.7	22	0.05	8.4	5.3	16	0.00	7.9	5
4_1200,1200	13.5	31	5.68	12.2	8.2	29	0.00	6.6	4
4_1300,1300	4.4	17	0.18	5.9	5.2	19	0.00	5.4	2
4_1400,1400	2.8	14	0.71	5.0	2.4	14	0.14	4.1	8
5_1500,1600	14.0	23	6.00	17.0	7.9	24	0.00	10.0	2
5_1600,1500	18.4	47	0.77	8.6	12.1	33	0.09	7.9	4
5_1600,1550	9.1	26	0.08	9.3	8.0	22	0.00	10.0	2
6_1820,1900	43.6	66	0.62	6.5	53.3	50	0.00	7.1	2
6_1900,1900	18.1	36	0.03	6.6	12.5	32	0.00	5.7	2
6_1900,1850	30.5	45	2.09	10.8	11.5	31	0.00	10.6	2
7_2250,2250	40.5	53	0.13	7.0	23.4	45	0.00	6.4	4
8_2600,2500	62.3	66	1.65	10.7	30.9	39	0.05	7.8	4
9_2800,2900	36.7	50	3.94	16.1	97.6	70	0.04	9.2	12
10_3100,3150	56.6	54	2.98	15.0	66.5	44	0.05	11.9	6

Table 7 β vector selection results for KC instances (Setting: Pricing cuts, aggressive region refinement)

	Con				Agg			
Instance	Time	It	Init	Tot	Time	It	Init	Tot
INST01	13.8	10	12.4	258.2	12.2	5	32.6	64.6
INST02	8.4	10	8.6	280.0	17.4	5	24.8	81.3
INST03	13.3	10	13.3	250.9	19.5	5	30.7	67.3
INST04	17.6	10	12.1	274.5	32.6	5	27.5	76.2
INST05	20.1	10	11.8	262.6	34.5	5	30.3	73.2
INST06	22.0	10	12.1	261.0	29.4	5	30.8	72.4
INST07	21.9	10	12.4	253.4	27.7	5	30.7	68.7
INST08	22.0	10	11.9	260.5	35.9	5	25.9	79.0
INST09	18.5	10	11.9	260.3	31.6	5	25.8	78.8
INST10	19.1	10	12.0	257.6	36.0	5	32.6	63.3
INST11	17.2	10	12.2	258.6	35.9	5	31.6	67.7
INST12	18.8	10	12.0	259.2	46.5	5	29.4	70.2

Table 8 Refinement results for small MMKP instances with $|\mathcal{K}| = 5$ (Setting: No pricing cuts, $\beta = \mathbf{1}$)