An algorithm for generating Lagrangian bound sets in Multiobjective Integer Programming

Mariagrazia Cairo[a,*], Lavinia Amorosi[a], Paolo Dell'Olmo[a], Serpil Sayın[b]

[a] *Department of Statistical Sciences, Sapienza University of Rome, P.le Aldo Moro 5, 00185 Rome, Italy*
{mariagrazia.cairo,lavinia.amorosi,paolo.dellolmo}@uniroma1.it
[b] *College of Administrative Sciences and Economics, Koç University, Sariyer, Istanbul 34450, Turkey*
ssayin@ku.edu.tr

Abstract

Lagrangian relaxation is a well-established technique for deriving strong bounds in single-objective discrete optimization. Its generalization to the multiobjective setting is not straightforward, as preserving the multiobjective structure leads to bound sets rather than scalar bounds. Recent studies show the existence of Lagrange multipliers that can yield tighter bound sets than those obtained from convex hull relaxations. However, from an algorithmic perspective, the potential benefits and challenges of applying Lagrangian relaxation to multiobjective integer programming remain largely unexplored. In this article, we develop the first algorithmic framework to generate bound sets for multiobjective integer linear programs via Lagrangian relaxation. Our approach preserves the multiobjective structure of the problem in the Lagrangian dual. Computational experiments on biobjective benchmark instances show that Lagrangian bound sets obtained by the algorithm are significantly tighter than those obtained from linear relaxations. In a proof of concept implementation, we embed our algorithm within a branch-and-bound scheme for a biobjective knapsack problem and assess the impact of Lagrangian bound sets on pruning efficiency and computational time in comparison with the linear programming bound sets. Our results illustrate the potential of Lagrangian bound sets for multiobjective integer programming.

Keywords: Multiobjective Discrete Optimization, Bound Sets, Lagrangian Relaxation, Branch-and-Bound Algorithm.

1. Introduction

Multiobjective Optimization (MOO) is concerned with simultaneous optimization of multiple conflicting objective functions over a feasible set. The solution of a (MOO) problem therefore consists of a set of points instead of a single point, referred to as the Pareto optimal or efficient set. When the feasible set of a (MOO) problem is a finite discrete set, the Pareto set is also finite.

During the last decade, significant progress has been made towards solving discrete (MOO) problems. Currently, scalarization-based algorithms, also referred to as objective

*Corresponding author

space approaches, seem to dominate the algorithmic landscape ([1, 2]). Scalarization refers to building a single objective subproblem based on the original (MOO) problem by using parameters. By systematically searching the space of all possible parameter values, the entire set of solutions can be recovered. The ϵ-constraint scalarization ([3, 4, 5]) and the Tchebycheff scalarization ([6, 7]) are the most commonly used for discrete (MOO) problems. The weighted sum scalarization may also be used; however, due to the non-convexity of the discrete (MOO) problems, only the solutions that lie on the boundary of the convex hull of the feasible set can be recovered this way ([8, 9]).

Multiobjective branch-and-bound approaches are generalizations of the single objective case ([10]). The primal and dual bounds in this setting are sets instead of single solutions and a comparison of these sets must be conducted for possible pruning of the nodes of a tree. Thus, the quality of the bound sets is critical for the success of these approaches. The branch-and-bound approach has the advantage of relating better to the decision space of a (MOO) problem, in contrast to search space algorithms that primarily explore the objective space. However, its computational performance is not yet competitive with objective space approaches ([11]).

In single objective discrete optimization, the Lagrangian relaxation method has emerged as a successful way of approaching some hard problems. Introduced in a formal way by Held and Karp ([12, 13]) to tackle the Traveling Salesperson Problem, this approach is based on the idea that some hard problems would become easier to solve when a set of complicating constraints is relaxed. This approach can be useful in providing bounds on the optimal value and guiding the search for near-optimal feasible solutions, whose quality can be evaluated using these bounds. Since Geoffrion's study ([14]), where the name Lagrangian relaxation has been introduced, several applications such as scheduling, resource allocation, unit commitment, facility location, and knapsack problems, have been addressed this way ([15, 16, 17, 18, 19, 20, 21]). These studies collectively demonstrate its potential as a viable alternative for tackling well-established problems.

One way to utilize Lagrangian relaxation in a (MOO) setting is to use it while solving a particular scalarization of the original problem. The use of weighted sum scalarization has been common. In [22], Ehrgott and Gandibleux resort to Lagrangian relaxation and subgradient optimization while solving a weighted sum scalarization of the biobjective traveling salesperson problem. In [23], Larson et al. define a convex combination of the objectives and relax the resulting single-objective problem. They illustrate their approach on a particular facility location problem for which a heuristic principle to derive primal bounds is also provided. In [24], it is also considered a scalarized formulation and Lagrangian relaxation in a single-objective setting is applied.

A general theoretical framework for duality in Multiobjective Integer Programs (MOIPs) has been given by Dunbar et al. in [25], where the authors provide an extension of Lagrangian duality to (MOIPs). They define a (multiobjective) Lagrangian dual problem and establish its theoretical properties and demonstrate that, with an appropriate choice of the Lagrange multipliers, a bound set that is tighter than the convex hull relaxation of the associated (MOIP) can be achieved. They also argue that better bound sets can be obtained even by a naïve enumeration of Lagrange multipliers. This is illustrated on two biobjective problems of relatively small size where, with a naïve grid-like enumeration of Lagrange multipliers, the Lagrangian relaxations can yield a better bound set than the one obtained by the convex

hull relaxation of the problem.

In [26], Brun et al. investigate the quality of the Lagrangian relaxation. They define the associated dual optimization problem, i.e., the optimization of the Lagrangian bound sets over the space of Lagrange multipliers for (MOIPs). They prove that a set of Lagrangian relaxations can generate bounds which are better than the ones provided by the convex hull relaxation. The authors illustrate on examples that specific relaxations can be tight at unsupported solutions; however, they do not discuss how such relaxations can be built in an algorithmic way.

The aim of the present study is to build upon the above recently established theoretical results and develop an algorithmic framework designed to generate bound sets for Multiobjective Integer Linear Programming (MOILP) problems via Lagrangian relaxation. We call such sets Lagrangian bound sets. We present a multiobjective dual descent framework in which the multiobjective nature of the Lagrangian dual optimization problem is maintained in a (MOILP) context. To the best of our knowledge, this constitutes the first such algorithmic attempt where several design choices, including the selection of search directions, stepsize and Lagrange multipliers updates, gap calculation and different ways of solving the relaxed multiobjective problem are investigated. We introduce six different rules for subgradient selection, as well as strategies for updating stepsizes and Lagrange multipliers and investigate their impact on the performance of the multiobjective dual descent algorithm. In our extensive computational experiments, we also compute the linear programming bound sets and assess them alongside Lagrangian bound sets with respect to the exact nondominated set of problem instances. The instances come from three different problem classes, the biobjective knapsack problem, uncapacitated facility location problem, and asymmetric traveling salesperson problem. We also integrate the Lagrangian bound sets into a state of the art branch-and-bound algorithm and assess their effectiveness on the number of generated nodes and the computational time, compared to the bound sets obtained by linear relaxation. The main contributions of this work can be summarized as follows:

- We propose the first general algorithmic framework for generating Lagrangian bound sets in multiobjective integer linear programming, helping to bridge the gap between existing theory and practical solution methods.
- We identify the key design elements and report on Lagrange multiplier update strategies with different subgradient and stepsize selection rules, and different strategies of solving the relaxed problems.
- We provide a computational assessment on different problem classes and a proof of concept integration within a state of the art branch-and-bound scheme, showing the practical potential of the approach.

The remainder of the paper is organized as follows. Section 2 introduces the relevant background, including basic definitions and the formulation of the single and multiobjective dual problems. Section 3 addresses the solution of the multiobjective Lagrangian dual, presenting the proposed algorithm and the associated Lagrange multiplier update procedure. Section 4 describes the considered problem classes and their relaxations. In Section 5 we evaluate the proposed algorithms on the test problems and we investigate their use within

a branch-and-bound algorithm with customized node selection, bound computation, and pruning mechanisms. Finally, conclusions and further researches are presented in Section 6.

2. Preliminaries

The general (MOILP) problem can be formulated as:

$$\begin{array}{ll} \max & Cx \\ \text{s.t.} & A^1x \leqq b^1, \\ & A^2x \leqq b^2, \\ & x \in \mathbb{Z}^n, \end{array} \tag{MOILP}$$

where $C \in \mathbb{R}^{p\times n}$, $A^1 \in \mathbb{R}^{m_1\times n}$, $A^2 \in \mathbb{R}^{m_2\times n}$, $b^1 \in \mathbb{R}^{m_1}$ and $b^2 \in \mathbb{R}^{m_2}$. In this formulation, the number of objective functions is given by the parameter p. The set $\mathcal{X} = \{x \in \mathbb{Z}^n : A^1x \leqq b^1,\ A^2x \leqq b^2\}$ represents the set of feasible solutions in the decision space, while $\mathcal{Y} = \{y \in \mathbb{R}^p : y = Cx : x \in \mathcal{X}\}$ is its image in the objective space. Since the objective function is vector-valued, the scalar concept of optimality fails, and we resort to the Pareto concept of optimality. We assume that $\mathcal{X} \neq \emptyset$ and finite.

Definition 1 (Dominance Relations)**.** *Let $y^1, y^2 \in \mathbb{R}^p$. It is said that:*

- *y^1 weakly dominates y^2 ($y^1 \geqq y^2$), if $y^1_k \geq y^2_k$ for $k = 1, \ldots, p$;*
- *y^1 strictly dominates y^2 ($y^1 > y^2$), if $y^1_k > y^2_k$ for $k = 1, \ldots, p$;*
- *y^1 dominates y^2 ($y^1 \geq y^2$), if $y^1_k \geq y^2_k$ and $y^1 \neq y^2$;*

Definition 2 (Efficient solution)**.** *A feasible solution $x^* \in \mathcal{X}$ is called efficient, or Pareto optimal, if there is no other solution $x \in \mathcal{X}$ such that $Cx \geq Cx^*$. The set of all efficient solutions is denoted by $\mathcal{X}_E$. Its corresponding image in the objective space is defined as $\mathcal{Y}_N = \{y \in \mathbb{R}^p : y = Cx : x \in \mathcal{X}_E\}$. Points belonging to $\mathcal{Y}_N$ are called nondominated points.*

Solving a (MOILP) problem means determining all the nondominated points $y \in \mathcal{Y}_N$ and for each nondominated point a corresponding solution $x \in \mathcal{X}$ such that $Cx = y \in \mathcal{Y}_N$, i.e. the minimal complete set of efficient solutions.

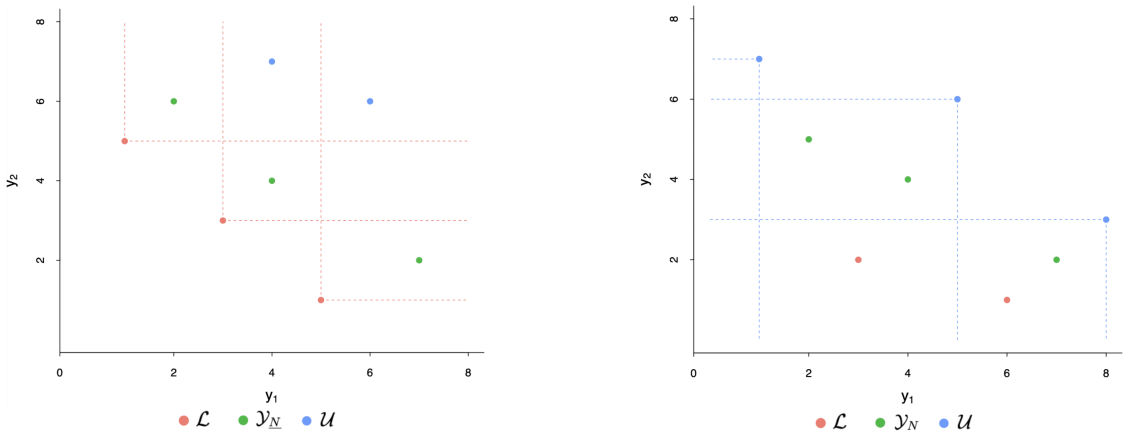
Following the notation in [27] we denote by $\mathcal{Q}_N$ the set of nondominated points of any set $\mathcal{Q} \subseteq \mathbb{R}^p$ with respect to componentwise maximization, by $\mathcal{Q}_{\underline{N}}$ the set of nondominated points of $\mathcal{Q}$ with respect to componentwise minimization, by $conv(\mathcal{Q})$ the convex hull of $\mathcal{Q}$, by $\text{cl}(\mathcal{Q})$ the closure of $\mathcal{Q}$ and by $\mathbb{R}^p_{\geqq} = \{y \in \mathbb{R}^p : y \geqq 0\}$. According to this notation, the set of supported nondominated points can be defined as follows: $\mathcal{Y}_{SN} = \{y \in \mathcal{Y}_N : y \in (conv(\mathcal{Y}) + \mathbb{R}^p_{\geqq})_N\}$.

In [22], bound sets are defined for a minimization problem. Below, they are adapted for a maximization problem.

Definition 3 (Bound sets)**.** *An upper bound set $\mathcal{U} \subseteq \mathbb{R}^p$ for $\mathcal{Y}_N$ is a $\mathbb{R}^p_{\geqq}$-closed (i.e., the set $\mathcal{U} - \mathbb{R}^p_{\geqq}$ is closed), $\mathbb{R}^p_{\geqq}$-bounded (i.e., there exists a $y \in \mathbb{R}^p$ such that $\mathcal{U} \subset y - \mathbb{R}^p_{\geqq}$) set with $\mathcal{U} = \mathcal{U}_N$ such that $\mathcal{Y}_N \subset (\mathcal{U} - \mathbb{R}^p_{\geqq})$. A lower bound set $\mathcal{L} \subseteq \mathbb{R}^p$ for $\mathcal{Y}_N$ is a $\mathbb{R}^p_{\geqq}$-closed, $\mathbb{R}^p_{\geqq}$-bounded set with and $\mathcal{L} = \mathcal{L}_N$ such that $\mathcal{Y}_N \subset cl(\mathbb{R}^p \setminus (\mathcal{L} - \mathbb{R}^p_{\geqq}))$.*

A lower bound set of $\mathcal{Y}_N$ for a maximization problem is a primal bound and is generally given by a set of known feasible points filtered by dominance. An upper bound set for $\mathcal{Y}_N$, on the other hand, is typically obtained by *relaxations*.

Figure 1 illustrates the difference in the definition of bound sets for minimization and maximization in a biobjective problem. The dashed lines in both cases illustrate how the dual bound set indicates where $\mathcal{Y}_N$ can potentially lie.



(a) Minimization problem

(b) Maximization problem

Figure 1: Illustration of bound sets in $\mathbb{R}^2$.

We use the one-sided Hausdorff distance in evaluating the bound sets.

Definition 4 (One-sided Hausdorff distance)**.** *Let $S_1, S_2 \subseteq \mathbb{R}^n$. The one-sided Hausdorff distance from S_1 to S_2 is defined as*

$$d(S_1, S_2) = \max_{s_1 \in S_1} \min_{s_2 \in S_2} \|s_1 - s_2\|. \tag{1}$$

where $\|\cdot\|$ represents the Euclidean norm.

Note that by construction (1) is not symmetric, i.e., in general $d(S_1, S_2) \neq d(S_2, S_1)$. For this reason, we will explicitly specify the role of the sets S_1, S_2 when using the one-sided Hausdorff distance throughout the paper.

2.1. Lagrangian duality in single objective optimization

A single objective integer linear programming (ILP) problem can be cast into our framework by setting $p = 1$ and $C \in \mathbb{R}^{1\times n}$ in (MOILP). Let z^* denote the optimal objective function value of this problem, i.e. $z^* = Cx^*$ where x^* is an optimal solution of (ILP). Assume that $A^1 x \leqq b^1$ are the hard constraints. Consider the following formulation:

$$\begin{aligned} \theta(\lambda) = \max \quad & Cx + \lambda^T (b^1 - A^1 x) \\ \text{s.t.} \quad & A^2 x \leqq b^2, \\ & x \in \mathbb{Z}^n, \end{aligned} \tag{LR_λ}$$

where $\lambda \in \mathbb{R}^{m_1}$ is a vector whose elements are referred to as Lagrange multipliers. Since the hard constraints are dropped from the set of constraints and appear in the objective function

with weights λ, in general, solving (LR_λ) is expected to be easier than solving the original problem. Moreover, with $\lambda \geqq 0$ and the feasible set of (LR_λ) containing the feasible set of (ILP), $\theta(\lambda)$ constitutes a relaxation of the original problem, i.e. $\theta(\lambda) \geq z^*$ for $\lambda \geqq 0$. The Lagrangian dual optimization problem (LD) then becomes:

$$\theta^* = \min_{\lambda \geqq 0} \theta(\lambda) \tag{LD}$$

where $\lambda \in \mathbb{R}^{m_1}$ denotes the dual variables. Let $\bar{\mathcal{X}} = \{x \in \mathbb{Z}^n | A^2 x \leqq b^2\}$ denote the feasible set of problem (LR_λ). It is known that solving the above Lagrangian dual problem to optimality corresponds to optimizing the original objective function over $conv(\bar{\mathcal{X}}) \cap \{x \in \mathbb{R}^n | A^1 x \leqq b^1\}$ [28]. Thus, the obtained solution has the potential to provide a better dual bound than the linear programming one, unless $conv(\mathcal{X}) = \{x \in \mathbb{R}^n : A^1 x \leqq b^1\}$ ([28], [26]). It is also known that $\theta(\lambda)$ is a convex piecewise linear function of λ ([28]), so the problem (LD) can be approached by a descent method. Since $\theta(\lambda)$ is not necessarily differentiable, its *subgradients* are typically employed and the resulting approach is referred to as a *subgradient algorithm*. A subgradient algorithm is essentially a dual descent algorithm that attempts to improve the dual bound using subgradients. It is an iterative procedure that updates the Lagrange multipliers by moving in the direction of a negative subgradient and a determined stepsize ([29]) with the goal of improving the dual bound.
The algorithm begins by choosing a starting solution $\lambda^0 \geqq 0$ and setting $k = 0$. Then, it solves the Lagrangian problem (LR_{λ^k}) to search for an optimal solution $x(\lambda^k)$. Since $b^1 - A^1 x(\lambda^k)$ constitutes a subgradient of $\theta(\lambda)$ at λ^k, it sets $\lambda^{k+1} = \max\{\lambda^k - \alpha^k(A^1 x(\lambda^k) - b^1), 0\}$, where α^k denotes the stepsize. Successively, it sets $k = k + 1$ and continues until a stopping condition is met ([28]).

2.2. Lagrangian duality in multiobjective optimization

With $p > 1$, again let $A^1 x \leqq b^1$ be the hard constraints. Define $\Lambda \in \mathbb{R}^{p \times m_1}$ as the matrix of Lagrange multipliers $\Lambda = (\lambda_1, \ldots, \lambda_p)$, where the r-th component λ_r contains the Lagrangian multipliers associated with objective r. The objective functions of the relaxed problem, which also has p objectives, are given by $L_r(x, \Lambda) = c_r x + \lambda_r(b^1 - A^1 x),\ r = 1, \ldots, p$. This can be expressed in vector notation as:

$$\theta(\Lambda) = \max_{x \in \bar{\mathcal{X}}} Cx + \Lambda(b^1 - A^1 x) \tag{MOLR$_\Lambda$}$$

where $\theta(\Lambda)$ denotes the nondominated set of this particular relaxation, which is a (MOILP) problem itself. We assume that $\bar{\mathcal{X}}$ is finite in order to simplify the description of the algorithm. We denote by $\bar{\mathcal{X}}_E^\Lambda$ the preimage of the nondominated set $\theta(\Lambda)$.

The following result ensures the validity of problem (MOLR_Λ) in accordance with the upper bound set definition we employ, which is slightly different than the one adopted in [25].

Proposition 2.1. *Let $\Lambda \in \mathbb{R}^{p \times m_1}_{\geqq}$. Then $\theta(\Lambda)$ is an upper bound set for $\mathcal{Y}_N$.*

Proof. By construction, the set $\theta(\Lambda)$ is a $\mathbb{R}^p_{\geqq}$-closed (it is a collection of discrete points), $\mathbb{R}^p_{\geqq}$-bounded ($\bar{\mathcal{X}}$ is finite) and $\theta(\Lambda) = \theta(\Lambda)_N$ ($\theta(\Lambda)$ is the nondominated set of problem

(MOLR$_\Lambda$)). Hence, it satisfies the first three properties of Definition 3. It remains to prove that $\mathcal{Y}_N \subset (\mathcal{U} - \mathbb{R}^p_{\leqq})$. Let $y^* \in \mathcal{Y}_N$. Then there exists $x^* \in \mathcal{X}$ such that $y^* = Cx^*$. Since x^* is feasible, we have $A^1 x \leqq b^1$. Moreover, since $\Lambda \geqq 0$, it follows that $\Lambda(b^1 - A^1 x) \geqq 0$. By the finiteness of $\bar{\mathcal{X}}$, and since $\mathcal{X} \subseteq \bar{\mathcal{X}}$, there exists $\hat{y} \in \theta(\Lambda)$ such that $\hat{y} \geqq Cx^* + \Lambda(b^1 - A^1 x^*) \geqq Cx^* = y^*$. Hence, there exists $r \in \mathbb{R}^p_{\geqq}$ such that $y^* = \hat{y} - r$, which implies $y^* \in \theta(\Lambda) - \mathbb{R}^p_{\geqq}$. It follows that $\mathcal{Y}_N \subseteq \theta(\Lambda) - \mathbb{R}^p_{\geqq}$. Since for a $\hat{y} \in \theta(\Lambda)$, $\forall s \in \mathbb{R}^p_{\geqq}$ and particularly $\forall s > 0$, $\hat{y} - s \in \theta(\Lambda) - \mathbb{R}^p_{\geqq}$, and $\hat{y} - s \notin \mathcal{Y}_N$, $\mathcal{Y}_N \subset \theta(\Lambda) - \mathbb{R}^p_{\geqq}$ follows.

□

In the spirit of the single objective case, the Lagrangian dual optimization problem seeks to identify the *best* nondominated set that can be achieved across all possible Λ values. In [25], this is captured by the set $\mathcal{Y}_{LD}$, which is described as the union of the nondominated sets of all possible Lagrangian relaxations of (MOILP) and the dual optimization problem is defined as identifying the minima over this set. The multiobjective Lagrangian dual problem then becomes:

$$\theta^* = \left(\bigcup_{\Lambda \in \mathbb{R}^{p \times m_1}_{\geqq}} \theta(\Lambda) \right)_{\underline{N}} \tag{MOLD}$$

In [25] it is also observed that, due to $\Lambda \in \mathbb{R}^{p \times m_1}_{\geqq}$, the set $\bigcup_{\Lambda \in \mathbb{R}^{p \times m_1}_{\geqq}} \theta(\Lambda)$ may be non-convex, disconnected, neither open nor closed, making problem (MOLD) difficult to solve. Thus obtaining θ^* explicitly may not be possible in general. The algorithm we present in Section 3 does not aim at constructing θ^* exactly. Instead it focuses on building an upper bound set $\mathcal{U} = \left(\bigcup_{\Lambda \in \mathcal{D}} \theta(\Lambda)\right)_{\underline{N}}$ along with a finite set of Lagrange multipliers, denoted as $\mathcal{D}$, actively contributing to the definition of $\mathcal{U}$.

3. Solving the Multiobjective Lagrangian Dual Optimization Problem

Solving problem (MOLD) exactly is a challenging task as is often the case for problem (LD) as well. Therefore our goal is to propose an algorithm that will deliver a solution to problem (MOLD) in a heuristic sense, generalizing the well-established approach for the single objective case that was briefly described in Section 2.1. The multiobjective nature of problem (MOLD) brings a number of additional questions that need to be addressed. The following example helps see some of these challenges.

Example 1. *Consider the following biobjective knapsack problem $\mathcal{P}$.*

$$\begin{aligned} \max \quad & \left(\sum_{i=1}^n c_{1i} x_i, \sum_{i=1}^n c_{2i} x_i \right) \\ \text{s.t.} \quad & \sum_{i=1}^n w_i x_i \leq W \\ & x_i \in \{0,1\} \qquad\qquad i \in \{1, \ldots, n\} \end{aligned} \tag{$\mathcal{P}$}$$

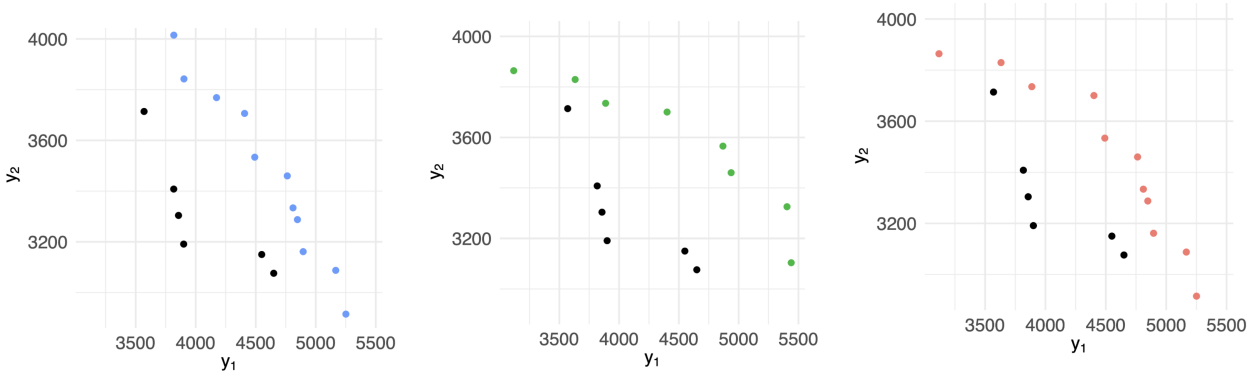
with $n = 10$ *and* $W = 2137$ *(see [4]), where:*

$$\begin{aligned} c_1 &= (62, 84, 977, 979, 874, 54, 269, 93, 881, 563), \\ c_2 &= (664, 982, 962, 140, 224, 215, 12, 869, 332, 537), \\ w &= (557, 898, 148, 63, 78, 964, 246, 662, 386, 272). \end{aligned}$$

With $\Lambda = (\lambda_1, \lambda_2)$, the biobjective problem $(BOLR_\Lambda)$ can be written as follows.

$$\begin{aligned} \max \quad & \left(\sum_{i=1}^{n} (c_{1i} - \lambda_1 w_i) x_i + \lambda_1 W, \sum_{i=1}^{n} (c_{2i} - \lambda_2 w_i) x_i + \lambda_2 W \right) \\ \text{s.t.} \quad & x_i \in \{0, 1\} \quad i \in \{1, \ldots, n\} \end{aligned} \tag{$BOLR_\Lambda$}$$

Figure 2a illustrates the nondominated set of problem $(BOLR_\Lambda)$ with $\Lambda^0 = (0.75, 0.75)$ and Figure 2b depicts the nondominated set of the relaxation with $\Lambda^1 = (0.95, 0.95)$.



(a) $\Lambda^0 = (0.75, 0.75)$ (b) $\Lambda^1 = (0.95, 0.95)$ (c) $\left(\bigcup_{\Lambda \in \{\Lambda^0, \Lambda^1\}} \theta(\Lambda)\right)_{\underline{N}}$

Figure 2: Comparison of Lagrangian bound sets with respect to the nondominated set for different values of Λ.

It can be observed that $\theta(\Lambda^0)$ provides a better bound for $\mathcal{Y}_N$ in the region with higher values of the first objective function whereas it is poor in the regions with higher values of the second objective. The reverse is observed in $\theta(\Lambda^1)$. When $\theta(\Lambda^0)$ and $\theta(\Lambda^1)$ are merged and filtered, the upper bound set improves as seen in Figure 2c. Note, however, that the Lagrangian bound set remains relatively poor at the central region. After we present the main flow of the algorithm, we discuss alternative ways of adjusting the Lagrange multipliers with the goal of improving the quality of the Lagrangian bound sets.

Algorithm 1 reports the pseudocode of the proposed algorithmic framework to calculate the bound sets $\mathcal{L}$ and $\mathcal{U}$ of a general (MOILP) problem. Along with the problem instance, the algorithm inputs a possibly empty lower bound set $\mathcal{L}$, initial Lagrange multipliers Λ^0, one of the rules we present in the next subsection to update the Lagrange multipliers, and stopping criteria.

The function call **SolveMOLR** indicates that the Lagrangian relaxation of the problem at iteration k, denoted as problem $(\text{MOLR}_{\Lambda^k})$ is built and solved for $\theta(\Lambda^k)$ along with its preimage $\bar{\mathcal{X}}_E^{\Lambda^k}$. For notational simplicity, we write $\bar{\mathcal{X}}_E^k$ instead of $\bar{\mathcal{X}}_E^{\Lambda^k}$. Then, for each

Algorithm 1 Solving problem MOLD

Input: MOILP problem $\mathcal{P}$ with p objectives, initial lower bound set $\mathcal{L}$, initial Lagrange multipliers Λ^0, Lagrange multipliers update rule $\mathcal{R}$, stopping criteria (threshold t, maximum number of iterations i_{max})

Output: Bound sets $\mathcal{L}$ and $\mathcal{U}$

Set $k \leftarrow 0$, $gap^k \leftarrow \infty$, $\mathcal{U} \leftarrow \emptyset$, $\mathcal{D} \leftarrow \emptyset$
while $gap^k > t$ **and** $k < i_{max}$ **do**
    $\theta(\Lambda^k) =$ **SolveMOLR**$(\mathcal{P}, \Lambda^k)$; let $\bar{\mathcal{X}}_E^k$ denote its preimage
    for $x \in \bar{\mathcal{X}}_E^k$ **do**
        $subg_x = b^1 - A^1 x$
        if $subg_x \geqq \mathbf{0}$ **then**
            $\mathcal{L} \leftarrow (\mathcal{L} \cup \{Cx\})_N$
        end if
    end for
    if $\left(\mathcal{U} \cup \theta(\Lambda^k)\right)_{\underline{N}} \neq \mathcal{U}$ **then**
        $\mathcal{U} \leftarrow (\mathcal{U} \cup \theta(\Lambda^k))_{\underline{N}}$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \Lambda^k$
    end if
    $(\Lambda^{k+1},\ gap^{k+1}) =$ **UpdateMultipliers**$(p, \Lambda^k, \mathcal{R}, r', \mathcal{L}, \mathcal{U}, \theta(\Lambda^k), \bar{\mathcal{X}}_E^k)$
/* $r' = 1 + (k \bmod p)$ if $\mathcal{R} =$*Cyclic Priority*, $r' = 0$ otherwise */
    $k \leftarrow k + 1$
end while
return $\mathcal{L}, \mathcal{U}$

solution $x \in \bar{\mathcal{X}}_E^k$, the associated subgradient $subg_x \in \mathbb{R}^{m_1}$ is calculated as the violation of the relaxed constraints, i.e. $subg_x = b^1 - A^1 x$ (line 5). If each component of the subgradient is greater than or equal to zero, x is feasible for the original problem $\mathcal{P}$, and its corresponding nondominated point $y = Cx$ is considered for inclusion in the lower bound set $\mathcal{L}$ (lines 6-7). Only points that are nondominated by existing elements in $\mathcal{L}$ are added, ensuring that the set remains minimal (line 7). Next, if $\theta(\Lambda^k)$ contains points that are not dominated by the current elements of the upper bound set $\mathcal{U}$ in a minimization sense, these are added to $\mathcal{U}$ and the corresponding Lagrange multipliers are added to the collection $\mathcal{D}$. The Lagrange multipliers are then updated using the selected rule according to the function described in Algorithm 2 where the gap is also calculated (line 14). The algorithm continues until the gap falls below the specified threshold or the iteration limit is reached.

Note that Algorithm 1 terminates after a finite number of iterations. This follows from the imposed upper bound on the number of iterations (line 2) and from the finiteness of $\bar{\mathcal{X}}$, which ensures that **SolveMOLR** can terminate in a finite number of steps each time it is called. We now proceed with the most crucial aspect of Algorithm 1, which is updating the Lagrange multipliers.

3.1. Updating the Lagrange Multipliers

In single-objective optimization, the problem defined at a particular iteration with Lagrange multipliers λ has one optimal value $\theta(\lambda)$, possibly produced by one or perhaps alternative optimal solutions. The algorithm can proceed by selecting a solution x and the subgradient associated with it. In a multiobjective setting, an iteration with the Lagrange multipliers Λ yields a nondominated set $\theta(\Lambda)$ with a preimage set $\bar{\mathcal{X}}_E^\Lambda$. Each element of $\bar{\mathcal{X}}_E^\Lambda$ has an associated subgradient. A subgradient, or perhaps a set of subgradients, that would contribute the most to the reduction of the *gap* between the Lagrangian bound set $\mathcal{U}$ and a lower bound set $\mathcal{L}$ would be ideal.

Algorithm 2 describes the function that updates the Lagrange multipliers according to six different rules. Since subgradients are directly related to solutions, the rules describe how one appropriate $x^* \in \bar{\mathcal{X}}_E^\Lambda$ is identified, with the exception of the first one which could identify up to p such solutions. From here on we denote the subgradient associated with a solution x as $subg_x = b^1 - A^1 x$.

Our first rule is the only one that involves distinct subgradients for each objective. The rest of the rules are designed to keep identical copies of the Lagrange multipliers associated with different objective functions and using a single subgradient to update them. This helps control the size of problem (MOLD) by reducing the number of variables from $p \times m_1$ to m_1.

1. *Priority rule.* Different Lagrange multipliers are maintained for each objective function. The rule identifies different subgradients and stepsizes, in a sense, as an immediate generalization of the single objective practice. The goal is to let the objective specific Lagrange multipliers evolve according to their own dynamics (line 3).

2. *Min rule.* By selecting a solution that is closest to the current primal bound set, the goal is to steer the search toward the regions where the Lagrangian bound set is already good (line 13).

Algorithm 2 Updating Multipliers

function UpdateMultipliers($p, \Lambda, \mathcal{R}, r', \mathcal{L}, \mathcal{U}, \theta(\Lambda), \bar{\mathcal{X}}_E^{\Lambda}$)
Input: number of objectives p, Lagrange multipliers Λ, Lagrange multipliers update rule $\mathcal{R}$, objective to be prioritized r', lower bound set $\mathcal{L}$, upper bound set $\mathcal{U}$, nondominated set of (MOLR$_\Lambda$) $\theta(\Lambda)$ and the associated efficient set $\bar{\mathcal{X}}_E^{\Lambda}$

if $\mathcal{R}$ =*Priority* **then**
  for $r = 1, \ldots, p$ **do**
    $y_r^* \in \arg\max_{y \in \theta(\Lambda)} y_r$; let x_r^* be its preimage
    $subg_{x_r} = subg_{x_r^*}$
    $\delta_r = \min_{y' \in \mathcal{L}}(y_r^* - y_r')$
    $\alpha_r = \gamma_r \dfrac{\delta_r}{\|subg_{x_r^*}\|^2}$
    $\lambda_r \leftarrow \max\{\lambda_r - \alpha_r\, subg_{x_r^*}, 0\}$
    $gap_r = \max_{u \in \mathcal{U}} \min_{l \in \mathcal{L}}(u_r - l_r)$
  end for
  $gap = \max_{r=1,\ldots,p}\ gap_r$
else
  if $\mathcal{R}$ =*Min* **then**
    $y^* \in \arg\min_{y \in \theta(\Lambda)} \min_{y' \in \mathcal{L}} \|y - y'\|$; let $x^* \in \bar{\mathcal{X}}_E^{\Lambda}$ be its preimage
    $\delta = \min_{y' \in \mathcal{L}} \|y^* - y'\|$
  end if
  if $\mathcal{R}$ =*Max* **then**
    $y^* \in \arg\max_{y \in \theta(\Lambda)} \max_{y' \in \mathcal{L}} \|y - y'\|$; let $x^* \in \bar{\mathcal{X}}_E^{\Lambda}$ be its preimage
    $\delta = \max_{y' \in \mathcal{L}} \|y^* - y'\|$
  end if
  if $\mathcal{R}$ =*Max-Min* **then**
    $y^* \in \arg\max_{y \in \theta(\Lambda)} \min_{y' \in \mathcal{L}} \|y - y'\|$; let $x^* \in \bar{\mathcal{X}}_E^{\Lambda}$ be its preimage
    $\delta = \min_{y' \in \mathcal{L}} \|y^* - y'\|$
  end if
  if $\mathcal{R}$ =*Cyclic Priority* **then**
    $y^* \in \arg\max_{y \in \theta(\Lambda)} y_{r'}$; let $x^* \in \bar{\mathcal{X}}_E^{\Lambda}$ be its preimage
    $\delta = \min_{y' \in \mathcal{L}} \|y^* - y'\|$
  end if
  if $\mathcal{R}$ =*Random* **then**
    $x^* = rand(x \in \bar{\mathcal{X}}_E^{\Lambda})$; let $y^* \in \theta(\Lambda)$ be its image
    $\delta = \min_{y' \in \mathcal{L}} \|y^* - y'\|$
  end if
  $subg_x = subg_{x^*}$
  $\alpha = \gamma \dfrac{\delta}{\|subg_{x^*}\|^2}$
  for $r = 1, \ldots, p$ **do**
    $\lambda_r \leftarrow \max\{\lambda_r - \alpha\, subg_{x^*}, 0\}$
  end for
  $gap = d(\mathcal{U}, \mathcal{L})$
end if
return (Λ, *gap*)

3. *Max rule.* The goal is to potentially move towards the unexplored regions of the objective space by selecting a solution that is farthest from the current primal bound set (line 17).

4. *Max–Min rule.* The goal is to identify the solution whose nearest element in the current primal bound set is farthest (line 21).

5. *Cyclic Priority Rule.* The goal is to emulate the priority rule by rotating the prioritized objective r' across iterations (line 25).

6. *Random rule.* A solution is selected as random. It serves as a baseline strategy with no structural preference (line 29).

In addition to the determination of a subgradient, an appropriate stepsize definition remains crucial to ensure an effective update of the Lagrange multipliers and to reduce the sensitivity of the algorithm to the initialization of Λ. Various stepsize rules have been proposed in the literature, including *exponentially decreasing stepsize*, *piecewise decreasing stepsize* and *Polyak step length* [30, 31, 32, 33, 34].

As it proved more effective in our preliminary tests, we use an adaptation of Polyak's stepsize given by:

$$\alpha = \gamma \frac{\delta}{\|subg_{x^*}\|^2}, \tag{2}$$

where $\gamma \in (0, 2)$ is a scaling parameter. In equation (2), x^* denotes the solution whose subgradient is selected, and δ denotes a rule-dependent measure of deviation between the image of the selected point, $y^* \in \theta(\Lambda)$ and the current primal bound set $\mathcal{L}$. The way this deviation is interpreted by different rules can be seen in Algorithm 2.

The scaling parameter γ is commonly initialized to 1.5 in the literature ([23]) and may be adjusted during the iterative process based on the evolution of a global gap measure between $\mathcal{U}$ and $\mathcal{L}$. This quantity is distinct from the local deviation measure δ used in equation (2) and is based on the one-sided Hausdorff distance given in Definition 4 (line 37).

Based on the above discussion, the Lagrange multipliers at iteration k are updated relying on the following formula:

$$\lambda_r^{k+1} = \max\{\lambda_r^k - \alpha_r^k subg_{x_r^*}, 0\} \tag{3}$$

where λ_r^k denotes the r^{th} row of Λ^k, $subg_{x_r^*}$ denotes the subgradient associated with objective r, and α_r^k denotes the associated stepsize at iteration k. Since the relaxed problem (MOLR_Λ) is defined for $\Lambda \geqq 0$, if $\lambda_r^k < 0$ we set it to $0 \in \mathbb{R}^{m_1}$. Except for the *Priority rule*, a single solution $x^* = x_r^*$ is identified and a single $\alpha^k = \alpha_r^k$ is calculated for $r \in \{1, \ldots, p\}$ throughout the algorithm. Thus, if initialized in the same way, the Lagrange multipliers λ_r^k remain identical for $r \in \{1, \ldots, p\}$. As mentioned earlier, adopting the same Lagrange multipliers for all the objectives reduces the dimension of problem (MOLD), which becomes more important as m_1 grows. Therefore, it is worth investigating the performance of the rules that implicitly promote a balanced exploration of the objective space by maintaining identical Lagrange multipliers for all objectives.

3.2. Solving problem (MOLR_Λ*)*

The problem (MOLR_Λ) must be solved at each iteration of the dual algorithm. Since this problem preserves its (MOILP) structure, its exact solution may turn out to be computationally expensive and may not pay off for the contribution it brings to the enumeration of the Lagrangian bound set $\mathcal{U}$. Therefore problem (MOLR_Λ) may be solved for a representative subset of $\theta(\Lambda)$. In our computational experiments, we compare two alternative solution strategies where we solve problem (MOLR_Λ) exactly or only for its supported nondominated points. At first glance, solving problem (MOLR_Λ) for its supported nondominated points instead of the entire set sounds counter-intuitive since the aim is to generate bound sets that improve upon those obtained from a convex hull relaxation. Supported nondominated points can be obtained via weighted sum scalarizations usually at a lower computational cost in comparision with an exact approach. Moreover, the scalarized problem has the same constraint set as the original relaxed problem (MOLR_Λ) and it may be possible to solve it via simple observations as discussed in Section 5.3 for some of the problem types we experiment with, leading to further computational savings.

4. Test Problems and Relaxations

To evaluate the performance of the proposed algorithm, we consider different standard classes of biobjective combinatorial optimization problems: the Biobjective Knapsack Problem (BOKP), the Biobjective Uncapacitated Facility Location Problem (BOUFLP), and the Biobjective Asymmetric Traveling Salesperson Problem (BOTSP). Indeed, these problems and their variants are widely used as benchmarks in the literature ([23, 35, 27, 4]).

Since Algorithm 1 is formulated for maximization problems, the (BOUFLP) and the (BOTSP) are reformulated as maximization problems. Below, we introduce the Lagrangian relaxations we work with. We note that alternative formulations and relaxations are possible for (BOUFLP) and (BOTSP).

4.1. Lagrangian Relaxation for (BOKP)

The Lagrangian relaxation of the (BOKP) can be written as follows:

$$\begin{array}{lll} \max & \displaystyle\sum_{i=1}^{n} c_{ri}x_i + \lambda_r \left(W - \sum_{i=1}^{n} w_i x_i \right), & r \in \{1,2\} \\ & x_i \in \{0,1\} & i \in \{1,\dots,n\} \end{array}$$

Here, n denotes the number of items. For each item $i \in \{1,\dots,n\}$, w_i is its weight and c_{ri} its profit with respect to the r-th objective. The binary decision variable x_i indicates whether item i is selected for the knapsack or not. The only constraint of the problem is given by the knapsack capacity W which is relaxed using the objective specific Lagrange multipliers $\lambda_r \in \mathbb{R}$, $r = 1, 2$.

4.2. Lagrangian Relaxation for (BOUFLP)

The Lagrangian relaxation of the (BOUFLP), reformulated as a maximization problem, can be written as follows:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m}\sum_{j=1}^{l} -c_{rij}x_{ij} + \sum_{j=1}^{l} -f_{rj}y_j \\
& + \sum_{i=1}^{m} \lambda_{ri}\left(1 - \sum_{j=1}^{l} x_{ij}\right), && r \in \{1,2\} \\
& x_{ij} \leqq y_j && i = 1,\ldots,m,\ j = 1,\ldots,l \\
& x_{ij} \in \{0,1\} && i = 1,\ldots,m,\ j = 1,\ldots,l \\
& y_j \in \{0,1\} && j = 1,\ldots,l
\end{aligned}
$$

Here, $\{1,\ldots,l\}$ denotes the set of potential facility locations and $\{1,\ldots,m\}$ is the set of customers. Opening a facility at location j and assigning customer i to location j incur positive costs with respect to the r-th objective, denoted by f_{rj} and c_{rij}, respectively. The binary variable y_j indicates whether a facility is opened at location j, and the binary assignment variable x_{ij} indicates whether customer i is assigned to the facility at location j. Customer i can be assigned to facility j only if the facility is open. The constraint set that is relaxed ensures that each customer is definitely assigned to some facility. $\lambda_r \in \mathbb{R}^m$, for $r \in \{1,2\}$ denotes the objective specific Lagrange multipliers associated with these assignment constraints. Since the relaxed constraint is an equality constraint, the associated Lagrange multipliers are unrestricted in sign.

4.3. Lagrangian Relaxation for (BOTSP)

The traveling salesperson problem seeks a tour of n cities where each city is visited exactly once at minimum cost. We refer to the flow formulation of the asymmetric version of the problem and we let $|N| = n$ and $|A| = m$ denote the sets of nodes and arcs. The binary variables y_{ij} indicate whether an arc belongs to the tour, and the variables x_{ij}, for $(i,j) \in A$, are the flow variables. Let $S(i)$ and $P(i)$ denote the set of successors and predecessors of node i respectively. According to this notation, the Lagrangian relaxation of the (BOTSP), reformulated as a maximization problem, can be rewritten as:

$$
\begin{aligned}
\max \quad & \sum_{(i,j)\in A} -c_{ij}y_{ij} + \sum_{i\in N} \delta_{ri}\left(1 - \sum_{(i,j)\in A} y_{ij}\right) \\
& + \sum_{j\in N} \mu_{rj}\left(1 - \sum_{(i,j)\in A} y_{ij}\right), && r \in \{1,2\} \\
& \sum_{k\in S(i)} x_{ik} - \sum_{h\in P(i)} x_{hi} = \begin{cases} n-1 & \text{if } i = 1 \\ -1 & \text{if } i \neq 1 \end{cases} && \forall i \in N \\
& x_{ij} \leqq (n-1)y_j && \forall (i,j) \in A \\
& x_{ij} \geqq 0 && \forall (i,j) \in A \\
& y_j \in \{0,1\} && \forall (i,j) \in A
\end{aligned}
$$

Since two different sets of constraints are relaxed, we introduce two distinct Lagrange multipliers, δ_r and μ_r. We can equivalently define the Lagrange multipliers $\lambda_r = (\delta_r, \mu_r)$ of dimension $2n$. The corresponding subgradient $subg_x \in \mathbb{R}^{2n}$ is therefore given by the vector

of constraint violations, which measures the extent to which each node has more or fewer than one incoming or outgoing arc.

5. Computational experiments

5.1. Research questions and computational setup

The primary goal of this computational study is to assess the effectiveness of the proposed algorithm as a means of computing upper bound sets for (MOILP) problems. To this end, we compare the quality of the Lagrangian bound sets to those obtained via linear programming relaxations. We explore the impact of the Lagrange multipliers update rules we presented in Section 3.1 on the quality of the resulting bound sets. We also study the implications of solving problem (MOLR_Λ) repeatedly in terms of computational effort since in essence this is a (MOILP) problem itself. Therefore, before we explore the Lagrange multiplier update rules, we investigate whether generating the complete nondominated set of each relaxed problem is necessary to ensure high-quality Lagrangian bound sets, or whether supported nondominated points alone are sufficient. By solving problems with different sizes and types, we want to evaluate if the overall effectiveness of the algorithm is influenced by problem type and by the number of primal or dual variables involved.

Algorithm 1 and 2 presented in Section 3 are implemented in Julia 1.11. Similarly, the Julia implementation of Bensolve, available at `https://github.com/kofgokhan/Bensolve.jl`, is used to solve the linear relaxation of the problem instances. To solve problem (MOLR_Λ) for the entire nondominated set, we use the *KirlikSayin* algorithm implimentation in the Multiobjective Julia library *MultiObjectiveAlgorithms.jl* at `https://github.com/jump-dev/MultiObjectiveAlgorithms.jl/tree/master` which is based on [4]. To solve problem (MOLR_Λ) for the supported nondominated set, we use the Aneja and Nair procedure ([36]). If the structure of the relaxed problem is *easy* as in the case of problems (BOKP) and (BOUFLP), we solve each scalarized problem by observation as explained in the relevant sections below. Otherwise, we resort to the *Dicothomy* algorithm available in *MultiObjectiveAlgorithms.jl* which is an implementation of the Aneja and Nair procedure ([36]).

We let $\mathcal{U}_{LP}$ denote the upper bound sets obtained via the linear relaxation, referred to as the LP bound sets, and let $\mathcal{U}^*_{LR}, \mathcal{U}_{LR}$ denote the Lagrangian bound sets obtained through a complete nondominated set or a supported nondominated set enumeration. We adopt this notation throughout the rest of this section.

In all experiments, the dual optimization algorithm is limited to a maximum of 100 iterations, corresponding to the maximum number of relaxed problems solved and the initial Lagrange multiplier Λ^0 is set to 0.

The value of the parameter γ involved in the stepsize calculation as defined in equation (2) is halved whenever the *gap* increases in a given iteration or remains invariant for 10 consecutive iterations since the last reduction ([23]). All the tests are performed on an HPC infrastructure equipped with 8 compute hosts running Linux, each equipped with 2 AMD Epyc 7452 processors, 64 compute cores and 256 GB RAM, for a total of 512 compute cores.

5.2. Experimental design

We evaluate the proposed algorithms on three classes of biobjective combinatorial optimization problems, using benchmark instances adopted in the literature. Whenever the library instances we work with have three objective functions, we delete the first one in order to obtain a biobjective problem. For the (BOKP), we consider the instances with the number of variables ranging in $\{10, 20, \ldots, 100\}$ introduced in [4]. For each problem size, a set of 10 instances are available. The (BOUFLP) instances come from two different sources. The series in [37] consists of 28 biobjective instances with 90 customers and 30 facilities. The instances proposed in [35] have an equal number of customers and facilities, taking values in $\{5, 6, 7, 8, 10, 12, 14\}$ and leading to a number of variables ranging in $\{30, 42, 56, 72, 110, 156, 210\}$ and they are defined according to different cost-generation rules. For each category 10 instances have been generated.

For the (BOTSP), we refer to the data in [38], with $\{10, 15, 20\}$ cities. For each problem size, 10 independent instances are available.

For all test instances the true nondominated set $\mathcal{Y}_N$ is available and all the bound sets are assessed with respect to it, using the following scaled one-sided Hausdorff distance as a quantitative indicator:

$$d(\mathcal{Y}_N, \mathcal{U}) = \frac{1}{\beta} \max_{y^* \in \mathcal{Y}_N} \min_{y' \in \mathcal{U}} \|y^* - y'\|. \tag{4}$$

The scaling parameter β is defined as the average of $\|y\|$ for $y \in \mathcal{Y}_N \cup \mathcal{U}$ in line with [25]. The goal is to make it less sensitive to outliers in the set $\mathcal{Y}_N \cup \mathcal{U}$, which could potentially happen due to the Lagrangian bound set $\mathcal{U}$.

For each nondominated point, the nearest element of the Lagrangian bound set is identified and the distance measure reports the worst such value. A small value of this measure indicates that every point of the true Pareto front is in close proximity of an element of $\mathcal{U}$.

The most extensive analysis is conducted on the biobjective knapsack problem. Relaxing the single constraint results in a single Lagrangian multiplier for each objective and an unconstrained Lagrangian relaxed problem, yielding a formulation that serves as a natural starting point for the analysis. Building on the insights gained from this analysis, we proceed to the other benchmark classes. There are increasingly more Lagrange multipliers in these classes; therefore, the dual optimization problem has a higher dimensionality. Moreover, the relaxed constraints are of equality type, leading to Lagrange multipliers that are unrestricted in sign.

5.3. Experimental Results

5.3.1. The Biobjective Knapsack Problem

We first address the impact of solving problem (MOLR_Λ) for the entire nondominated set or for the supported nondominated set. Since the relaxed (BOKP) has an *easy* structure, we solve each scalarized problem encountered in the Aneja Nair procedure by observation. Since the relaxed problem is unconstrained over binary variables, we set each x_i to 1 if $p_1(c_{1i} - \lambda_1 w_i) + p_2(c_{2i} - \lambda_2 w_i) > 0$, where p_1 and p_2 are the scalarization coefficients; otherwise, $x_i = 0$, for $i \in \{1, \ldots, n\}$. In this experiment, we fix the Lagrange multipliers update rule as the *Max-Min rule*.

In Table 1, for each instance size (number of variables), we report the corresponding average distance as well as the average percentage improvement to the linear relaxation. The

percentage improvement of the Lagrangian relaxations with respect to the linear relaxation is defined as:

$$\%\text{IMP} = \frac{d(\mathcal{Y}_N, \mathcal{U}_{LP}) - d(\mathcal{Y}_N, \mathcal{U}_{LR})}{d(\mathcal{Y}_N, \mathcal{U}_{LP})} \times 100$$

n	**Linear Relaxation**	**Lagrangian Relaxation**					
	Bensolve	Complete search			Linear scalarized search		
	$d(\mathcal{Y}_N, \mathcal{U}_{LP})$	$d(\mathcal{Y}_N, \mathcal{U}^*_{LR})$	%IMP	Time	$d(\mathcal{Y}_N, \mathcal{U}_{LR})$	%IMP	Time
10	0.0879	0.0532	39%	17.3	0.0602	32%	0.7
20	0.0585	0.0441	25%	40.8	0.0470	20%	0.8
30	0.0281	0.0200	29%	56.2	0.0246	12%	1.0
40	0.0258	0.0126	51%	170.9	0.0161	38%	1.4
50	0.0240	0.0126	48%	127.0	0.0141	41%	1.4
60	0.0189	0.0108	43%	203.3	0.0115	39%	2.1
70	0.0185	0.0078	58%	659.2	0.0081	56%	2.1
80	0.0098	0.0053	46%	325.6	0.0068	31%	2.6
90	0.0122	0.0045	63%	1875.9	0.0063	49%	3.1
100	0.0093	0.0038	59%	717.1	0.0055	40%	3.4

Table 1: Average distances from the nondominated set to the linear (Bensolve) and the Lagrangian (Complete search, linear scalarized search) bound sets for different problem sizes (n) of the (BOKP). Average percentage improvements and computational time (seconds) are also reported.

The results show that the Lagrangian bound sets outperform the LP bound sets in terms of quality. Indeed, non-aggregated results suggest that, in 99 out of 100 instances, the Lagrangian bound sets obtained with the exact nondominated sets improve upon the LP bound sets. In the Lagrangian bound sets obtained via supported nondominated sets this occurs in 91 out of 100 instances. While we observe that the exact nondominated set approach (Complete search) outperforms the supported nondominated approach (Linear scalarized search), this comes at a considerable computational cost. Indeed, Table 1 shows that the computational time required by the complete search may become overwhelming with increasing problem size. As a consequence, in the remainder of the computational experiments, we maintain the linear scalarized approach and solve the Lagrangian relaxations only for their supported nondominated points.

We summarize in Table 2 the average results associated with the different Lagrange multipliers update rules introduced in Section 3.1. The results suggest that the *Priority rule* is the most effective among all. On average, the introduction of two distinct sets of parameters, as opposed to a single one, leads to an improvement in bound quality of about 10%, with maximum gains of nearly 20% observed for instances with 20 and 50 variables.

Regarding the rules that involve the selection of a unique subgradient, on average, the *Max-min rule* provides the best performance across instances. Although other rules may outperform it on specific sizes, they tend to perform worse overall. The only exception is the *Cyclic Priority rule*, which tends to approach *Max-min* performance, likely because its alternation of first and last values across iterations partially mimics the *Max-min* logic.

n	Lagrange multipliers update rules					
	Priority	*Min*	*Max*	*Max-Min*	*Cyclic Priority*	*Random*
10	0.0590	0.0643	0.0586	0.0602	0.0624	0.0626
20	0.0378	0.0491	0.0486	0.0470	0.0468	0.0474
30	0.0215	0.0263	0.0262	0.0246	0.0244	0.0249
40	0.0161	0.0185	0.0163	0.0161	0.0159	0.0166
50	0.0114	0.0154	0.0156	0.0141	0.0140	0.0155
60	0.0090	0.0138	0.0162	0.0115	0.0115	0.0116
70	0.0081	0.0082	0.0084	0.0081	0.0083	0.0080
80	0.0064	0.0069	0.0081	0.0068	0.0068	0.0065
90	0.0056	0.0102	0.0062	0.0063	0.0063	0.0059
100	0.0051	0.0077	0.0062	0.0055	0.0061	0.0059
Avg	0.0180	0.0220	0.0210	0.0200	0.0203	0.0205

Table 2: Average distances from the nondominated set to the Lagrangian bound sets for different problem sizes (n) of the (BOKP) and different Lagrange multipliers update rules. The average among different sizes is also reported.

Since the *Priority rule* outperforms all alternatives, we use this rule to compute the Lagrangian bound sets in the following experiments.

Finally, Table 3 reports summary statistics on the distance from the nondominated sets to sets computed via linear and Lagrangian relaxations for different problem sizes. The percentage of instances in which the Lagrangian relaxation outperforms the linear relaxation (%LR $>$ LP) is also reported. This percentage is computed according to the proportion of instances for which the Lagrangian relaxation provides better bound sets than the LP relaxation. The percentage improvement, reported under the header %IMP, is computed over instances for which the Lagrangian relaxation outperforms the linear relaxation.

On average, it can be said that the Lagrangian bound sets outperform the LP bound sets in (BOKP) instances. The difference appears to be more prominent as the problem size increases.

5.3.2. The Biobjective Uncapacitated Facility Location Problem

The (BOUFLP) is a widely studied benchmark in the literature, characterized by assignment constraints whose relaxation results in m Lagrange multipliers for each objective. Thus problem (MOLD) is higher dimensional and more challenging. Another difference from the (BOKP) problem is the unconstrained nature of problem (MOLD) since equality constraints are relaxed. As stated before, we use the *Priority rule* for Lagrange multipliers updates and solve problem (MOLR$_\Lambda$) for its supported nondominated points. This relaxation also has an *easy* structure and we can solve the weighted sum problems of the Aneja and Nair procedure by observation. For a pair of scalarization coefficients (p_1, p_2), we define the weighted assignment and opening costs as $\tilde{c}_{ij} = p_1(c_{1ij} - \lambda_{1i}) + p_2(c_{2ij} - \lambda_{2i}), i \in \{1, \ldots, m\}, j \in \{1, \ldots, l\}$, and $\tilde{f}_j = p_1 f_{1j} + p_2 f_{2j}, \forall j \in \{1, \ldots, l\}$. The relaxation has only one set of constraints in

n	Linear Relaxation $d(\mathcal{Y}_N, \mathcal{U}_{LP})$	Lagrangian Relaxation $d(\mathcal{Y}_N, \mathcal{U}_{LR})$	%(LR > LP)	%IMP
10	0.0879 [0.0319, 0.2242]	0.0590 [0.0317, 0.1014]	70%	33%
20	0.0585 [0.0331, 0.1685]	0.0378 [0.0243, 0.0624]	90%	35%
30	0.0281 [0.0196, 0.0455]	0.0215 [0.0081, 0.0304]	90%	23%
40	0.0258 [0.0174, 0.0587]	0.0161 [0.0103, 0.0219]	90%	38%
50	0.0240 [0.0114, 0.0426]	0.0114 [0.0080, 0.0144]	100%	53%
60	0.0189 [0.0093, 0.0352]	0.0090 [0.0055, 0.0129]	100%	53%
70	0.0185 [0.0087, 0.0484]	0.0081 [0.0055, 0.0133]	100%	56%
80	0.0098 [0.0075, 0.0116]	0.0064 [0.0046, 0.0090]	90%	35%
90	0.0122 [0.0068, 0.0279]	0.0056 [0.0041, 0.0073]	100%	54%
100	0.0093 [0.0065, 0.0142]	0.0051 [0.0037, 0.0064]	100%	46%

Table 3: Average, minimum and maximum distances from the nondominated set to the linear and Lagrangian bound sets for different problem sizes (n) of (BOKP) instances. The percentage of instances in which the Lagrangian relaxation outperforms the linear one and their average percentage improvements are also reported.

addition to binary restrictions on both x_{ij} and y_j variables, given by

$$x_{ij} \leqq y_j \quad i = 1, \ldots, m, \; j = 1, \ldots, l.$$

Thus a facility j would be opened only if the sum of the positive assignment contributions outweigh the fixed cost. We compute $C_j = \sum_{i=1}^{m} \max\{0, \tilde{c}_{ij}\} + \tilde{f}_j$. If $C_j > 0$, facility j is opened and all customers i such that $\tilde{c}_{ij} > 0$ are assigned to it. Otherwise, facility j remains closed and no assignment is performed.

Instance set	Linear Relaxation $d(\mathcal{Y}_N, \mathcal{U}_{LP})$	Lagrangian Relaxation $d(\mathcal{Y}_N, \mathcal{U}_{LR})$	%(LR > LP)	%IMP
Fernandez	0.2313 [0.0000, 0.5915]	0.0799 [0.0029, 0.1857]	79%	70%
Forget (all)	0.0899 [0.0055, 0.2048]	0.0932 [0.0163, 0.2305]	46%	18%
Forget (easy)	0.2313 [0.0464, 0.1655]	0.0799 [0.0707, 0.1254]	67%	20%

Table 4: Average, minimum and maximum distances from the nondominated set to linear and Lagrangian bound sets across different instance classes (Instace set) of (BOUFLP). The percentage of instances in which the Lagrangian relaxation outperforms the linear one and their average percentage improvements are also reported.

Table 4 reports the results for the F-series instances introduced in [37] and (UFLP) instances from [35] where it is observed that the performance of the Lagrangian bound sets considerably depends on the data set. In 22 out of 28 the *Fernandez* instances, (79%), the Lagrangian bound sets are better than the LP bound sets. For two instances in this set, the distance from the nondominated set to the LP bound set is zero. In the remaining cases, the

quality of Lagrangian bound sets worsens, with a moderate deterioration (around 26%) in three instances and a larger one (74%) in one instance. Thus the overall strong performance is based on the instances where the LP bound set is weak. It is worth noting that these instances are characterized by heterogeneous cost ranges, with facility opening costs being higher than assignment costs.

We consider additional (UFLP) instances from [35], which are characterized by an equal number of customers and facilities and by different data generation schemes. In these instances, assignment costs and facility opening costs are generated within distinct ranges, leading to heterogeneous problem structures. In [35], the instances are labeled as *easy* and *hard* depending on the range of facility opening costs: higher opening costs typically yield easier instances, whereas lower opening costs result in more challenging ones. Our computational results show that, while the Lagrangian bound sets display improvements over the LP bounds in some cases, this is not consistent through different instance classes. The Lagrangian bound sets are weaker on instances with *hard* cost structures, confirming the observations reported in [35]. Overall, an improvement over the linear relaxation is achieved in approximately 46% of the instances. In about 7% of all the instances, the linear bound sets already matches the nondominated sets; moreover, the quality of the Lagrangian bound sets is on average about 19% worse than the linear ones across all of them. Interestingly, when restricting the analysis to instances characterized by more homogeneous cost ranges, the percentage of instances which improves upon the linear relaxation increases to about 67%.

5.3.3. The Biobjective Traveling Salesperson Problem

The (BOTSP) comes with a rich formulation and the choice of the constraints to be relaxed plays a crucial role. Relaxing the degree constraints allows the Lagrange multipliers to directly act on the arc selection variables, which are those defining the objective functions. This leads to a meaningful penalization of infeasible solutions and, consequently, to stronger bounds. An alternative relaxation based on the flow conservation constraints is also possible. Our preliminary experiments with this approach resulted in significantly weaker Lagrangian bound sets. For this reason, in the following, we refer to the Lagrangian relaxation introduced in Section 4.3. Differently from the previous cases the relaxed problem has not an *easy* structure and we can not solve it by observation.

Table 5 reports the performance of our algorithm for the (BOTSP) instances in [38]. The results show that the Lagrangian bound sets improve upon the LP bound sets for smaller instances, with a significant gain for $n = 10$ and a more moderate improvement for $n = 15$. However, a moderate improvement is observed only in the 30% of the larger instances with $n = 20$. In fact, while the quality of the Lagrangian bound sets remains stable across different problem sizes, the quality of the LP bound sets improves with increasing problem size. For this reason the quality of the Lagrangian bound sets is on average about 19% worse than the linear ones across all instances for $n = 20$.

Overall, these results indicate that the Lagrangian relaxation can lead to improved bounds in several cases, although its effectiveness may vary across instance sizes.

n	Linear Relaxation $d(\mathcal{Y}_N, \mathcal{U}_{LP})$	Lagrangian Relaxation $d(\mathcal{Y}_N, \mathcal{U}_{LR})$	%(LR > LP)	%IMP
10	0.1509 [0.0917, 0.2114]	0.0977 [0.0587, 0.1571]	80%	42%
15	0.1413 [0.0837, 0.3230]	0.1135 [0.0794, 0.1335]	60%	28%
20	0.1010 [0.0476, 0.2145]	0.1197 [0.1007, 0.1463]	30%	21%

Table 5: Average, minimum and maximum distances from the nondominated set to linear and Lagrangian bound sets for different problem sizes (n) of asymmetric (BOTSP) instances. The percentage of instances in which the Lagrangian relaxation outperforms the linear one and their average percentage improvements are also reported.

5.3.4. Preliminary Branch-and-Bound Results: A Proof of Concept

Motivated by the results obtained across the different problem classes, which suggest that the Lagrangian relaxation can provide tighter bounds in several settings, we investigate its use within a branch-and-bound framework. The goal of this analysis is not to develop a fully optimized algorithm, but rather to provide a proof of concept and assess the potential of embedding the Lagrangian bound sets within an exact solution approach. The implementation of the branch-and-bound algorithm builds upon the code available at `https://github.com/SuneGadegaard/RAMOO2024`. The original Python implementation, designed for minimization problems, has been implemented in Julia and adapted to solve maximization problems.

From a methodological perspective, the overall structure of the algorithm follows the standard components of a branch-and-bound scheme, namely node selection, branching, bound computation, and pruning. Node selection is performed according to a best-bound strategy, i.e. the node with the most promising bound is selected for exploration in both linear and Lagrangian settings.

Regarding branching, in the linear relaxation framework, variables are typically selected based on their fractional values. However, in the Lagrangian setting, integrality is preserved, and this criterion is no longer meaningful. For this reason, we adopt a simple random selection strategy in both cases, while noting that more sophisticated rules, exploiting information from the dual algorithm, could be investigated.

As for bound computation, we use either the linear or the Lagrangian relaxation. The linear relaxation is solved using a standard LP solver by first computing the extreme solutions via lexicographic optimization and then iteratively generating additional supported nondominated points through weighted-sum scalarizations. To compute the Lagrangian bound sets, we resort to Algorithm 1 and adopt the algorithmic configurations that yielded the best performance in the previous experiments. Specifically, we consider two settings: the first is based on the *Priority rule* and distinct Lagrange multipliers are defined for different objectives, while the second is based on the same Lagrange multipliers for all of them combined with the *Max-Min rule*. Interestingly, contrary to what was observed in previous sections, the *Max-Min rule* strategy appears to be more effective within a branch-and-bound framework. This difference stems from the fact that the previous analysis only considers the root node of a branch-and-bound tree, i.e., a single problem without variable fixings, whereas a proper branch-and-bound framework involves subproblems at increasing depths.

In this context, the *Priority rule* tends to drive the search toward extreme regions, which combined with the increasing variable fixings at deeper nodes, may lead to solutions that are more likely to be infeasible with respect to the original problem and thus farther from the Pareto front. In contrast, the *Max-Min rule* exhibits a more balanced behavior, promoting a trade-off between intensification and diversification, as the search is guided by points that maximize their minimum distance from the current primal bound.

Finally, there are differences in the pruning mechanism between the two settings. When LP bound sets are used, the nodes can be typically pruned by optimality, infeasibility, and bounding. With the Lagrangian bound sets, optimality cannot be used as a pruning criterion anymore. With LP bound sets, indeed, a node can be pruned if the enumerated nondominated set is a singleton having integer coordinates. This reasoning fails when applied to the Lagrangian case, since the existence of a unique integer feasible solution to the relaxed problem does not imply optimality for the original problem, as it depends on a specific choice of Lagrange multipliers. It follows that different Lagrange multipliers (not necessarily attained by the procedure) may lead to different solutions, potentially yielding a set of candidate points. Pruning by infeasibility remains theoretically possible, but less likely. Since the Lagrangian relaxation is defined over a larger feasible set with respect to the original problem ($\mathcal{X} \subset \bar{\mathcal{X}}$), a node can be pruned by infeasibility only if it violates the constraints that are not relaxed; in that case, it is infeasible for the original problem as well. In the specific case of the knapsack problem, considered in this implementation, pruning by infeasibility does not occur. Indeed, every solution is feasible for the relaxed problem once the only problem constraint has been relaxed. Pruning by bound remains unchanged in principle for the Lagrangian bound sets.

The results reported in Table 6 refer to the use of a branch-and-bound algorithm to solve instances of the (BOKP). A time limit of 1800 seconds has been imposed for each run. The bound sets are computed using the *Max-Min rule*. Nevertheless, given the good performance of the *Priority rule* at the root node, hybrid strategies could be considered, where different Lagrange multipliers update rules are applied at different depths of the tree.

Although preliminary, the results obtained already highlight some interesting features. The Lagrangian relaxation consistently leads to a reduction in the number of generated nodes and, especially for certain sizes of instances, also yields competitive computational times compared to the linear relaxation. Indeed, in terms of computational time, it is competitive on small instances and improves as the problem size increases, with the best results on larger ones. When the branch-and-bound algorithm is terminated due to the time limit, the Lagrangian version seems to deliver a better nondominated set than the linear one. For problems with 80 variables, the Lagrangian version computes 65% of the nondominated points whereas the linear one reaches 43%. For problems with 90˘100 variables where most runs hit the time limit, the Lagrangian version is able to compute 51% of the nondominated points while the linear version reaches only 8%.

6. Conclusion

This paper investigates the use of Lagrangian relaxation in a multiobjective integer programming setting, with the aim of assessing its effectiveness in bounding the nondominated set. To this end, we propose an algorithmic framework for generating Lagrangian bound

n	Linear Relaxation			Lagrangian Relaxation		
	AvgTime	AvgNodes	TimeLimit(%)	AvgTime	AvgNodes	TimeLimit(%)
10	4.45	320.4	0%	1.90	264.4	0%
20	4.85	3174.4	0%	3.76	2299.8	0%
30	7.51	12873.6	0%	9.48	7632.0	0%
40	25.24	57969.8	0%	34.14	28313.2	0%
50	39.61	81137.0	0%	56.41	43729.2	0%
60	174.73	253679.0	0%	200.09	124124.8	0%
70	545.51	538526.8	0%	311.90	317839.4	20%
80	1016.85	791636.0	10%	893.14	349705.0	30%
90	391.01	805380.8	80%	769.51	310083.0	70%
100	-	825523.0	100%	934.02	257031.4	90%

Table 6: Average computational performance (time (seconds) and # of nodes) of the branch-and-bound algorithm with linear and Lagrangian relaxations for (BOKP) instances of different sizes. Reported times exclude instances that reached the time limit. A dash (–) indicates no runs completed within the time limit on the class instances. The percentage of instances that hits the time limit is also reported.

sets, providing a first step towards bridging the gap between existing theoretical results and practical solution methods. The computational results show that Lagrangian bound sets can improve upon the linear ones across different problem classes, highlighting the potential of the proposed framework. However, their effectiveness depends on the structure of the Lagrangian relaxation, the instance characteristics, and the algorithmic choices. Our results suggest that the theoretically established benefits can be effectively realized within a general multiobjective algorithmic framework.

At the same time, our findings indicate that the performance of the method depends on several components, offering insight into the behavior of the approach and suggesting directions for further improvement. An immediate step in this direction concerns the design of problem-specific Lagrangian heuristic approaches with a component that *repairs* the solutions obtained by the Lagrangian relaxation to feasibility, thereby improving the quality of the lower bound set. A better lower bound set could lead to more effective multiplier updates. Customized Lagrangian methods that deliver high quality lower and upper bound sets may provide heuristic solutions with an associated gap to challenging (MOILP) problems.

We observe that the choice of the underlying formulation and of the constraints to be relaxed play a crucial role for both obtaining strong bounds and for solving the relaxed problem efficiently. For instance, for the symmetric (BOTSP) problem, a formulation that leads to a Lagrangian relaxation with a spanning tree structure may be more effective. Such problems and formulations can be explored in depth in future work.

At a higher level, an important research direction is to investigate how Lagrangian relaxation can be exploited within different solution paradigms. The initial branch-and-bound implementation presented here provides encouraging results. Further developments may include the design of more informed branching rules exploiting dual information and the investigation of hybrid update strategies within the search process. Likewise, lower and upper bound sets obtained via Lagrangian heuristics may be used in reducing the search space of

exact methods.

This work addresses some of the challenges associated with developing subgradient-based multiobjective dual algorithms. More sophisticated techniques such as bundle methods could be explored in the multiobjective context. Finally, evaluating the approach on more realistic instances and exploring its performance on problems with more than two objectives represent directions for future research.

Data and code availability statement

Data and code are available from the authors upon request.

Acknowledgments

This paper originated during a one-month stay of Serpil Sayın at Sapienza University of Rome. Further research to complete it has also been supported by Sapienza University of Rome via project no. AR125199C2F8C468.

References

[1] K. Dächert, T. Fleuren, K. Klamroth, A simple, efficient and versatile objective space algorithm for multiobjective integer programming, Mathematical Methods of Operations Research 100 (1) (2024) 351–384.

[2] S. Tamby, D. Vanderpooten, Enumeration of the nondominated set of multiobjective discrete optimization problems, INFORMS Journal on Computing 33 (1) (2021) 72–85. doi:10.1287/ijoc.2020.0953.

[3] Y. Haimes, On a bicriterion formulation of the problems of integrated system identification and system optimization, IEEE transactions on systems, man, and cybernetics (3) (1971) 296–297.

[4] G. Kirlik, S. Sayın, A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems, European Journal of Operational Research 232 (3) (2014) 479–488. doi:https://doi.org/10.1016/j.ejor.2013.08.001.
URL https://www.sciencedirect.com/science/article/pii/S0377221713006474

[5] M. Mesquita-Cunha, J. R. Figueira, A. P. Barbosa-Póvoa, New ϵ-constraint methods for multi-objective integer linear programming: A pareto front representation approach, European Journal of Operational Research 306 (1) (2023) 286–307. doi:https://doi.org/10.1016/j.ejor.2022.07.044.
URL https://www.sciencedirect.com/science/article/pii/S0377221722006142

[6] P. Eswaran, A. Ravindran, H. Moskowitz, Algorithms for nonlinear integer bicriterion problems, Journal of Optimization Theory and Applications 63 (2) (1989) 261–279.

[7] T. Holzmann, J. C. Smith, Solving discrete multi-objective optimization problems using modified augmented weighted tchebychev scalarizations, European Journal of Operational Research 271 (2) (2018) 436–449.

[8] M. Ehrgott, Multicriteria optimization, Springer Books (2005).

[9] S. Helfrich, T. Perini, P. Halffmann, N. Boland, S. Ruzika, Analysis of the weighted tchebycheff weight set decomposition for multiobjective discrete optimization problems, Journal of Global Optimization 86 (2) (2023) 417–440.

[10] A. Przybylski, X. Gandibleux, Multi-objective branch and bound, European Journal of Operational Research 260 (3) (2017) 856–872. doi:https://doi.org/10.1016/j.ejor.2017.01.032.
URL https://www.sciencedirect.com/science/article/pii/S037722171730067X

[11] N. Forget, Solution algorithms for multi-objective integer linear programming models: A study of the branch-and-bound algorithm applied to the multi-objective case: Phd dissertation, Ph.D. thesis, Aarhus, Denmark (2022).

[12] M. Held, R. M. Karp, The traveling-salesman problem and minimum spanning trees, Operations Research 18 (6) (1970) 1138–1162. doi:10.1287/opre.18.6.1138.

[13] M. Held, R. M. Karp, The traveling-salesman problem and minimum spanning trees: Part ii, Math. Program. 1 (1) (1971) 6–25. doi:10.1007/BF01584070.

[14] A. Geoffrion, Lagrangian relaxation and its uses in integer programming, Mathematical Programming 2 (01 1974).

[15] M. L. Fisher, Optimal solution of scheduling problems using lagrange multipliers: Part i, Operations Research 21 (5) (1973) 1114–1127. doi:10.1287/opre.21.5.1114.

[16] J. F. Shapiro, Generalized lagrange multipliers in integer programming, Operations Research 19 (1) (1971) 68–76. doi:10.1287/opre.19.1.68.

[17] M. L. Fisher, J. F. Shapiro, Constructive duality in integer programming, Siam Journal on Applied Mathematics 27 (1974) 31–52.
URL https://api.semanticscholar.org/CorpusID:18142745

[18] H. Everett, Generalized lagrange multiplier method for solving problems of optimum allocation of resources, Operations Research 11 (3) (1963) 399–417. doi:https://doi.org/10.1287/opre.11.3.399.

[19] C. Lemaréchal, Lagrangian Relaxation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 112–156. doi:10.1007/3-540-45586-8_4.

[20] M. B. Rosenwein, Discrete location theory, edited by p. b. mirchandani and r. l. francis, john wiley & sons, new york, 1990, 555 pp, Networks 24 (1994) 124–125.
URL https://api.semanticscholar.org/CorpusID:206313877

[21] S. Martello, P. Toth, Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., USA, 1990.

[22] M. Ehrgott, X. Gandibleux, Bound sets for biobjective combinatorial optimization problems, Computers & Operations Research 34 (9) (2007) 2674–2694. doi:https://doi.org/10.1016/j.cor.2005.10.003.

[23] T. Larsson, N.-H. Quttineh, I. Åkerholm, A lagrangian bounding and heuristic principle for bi-objective discrete optimization, Operational Research 24 (03 2024). doi:10.1007/s12351-024-00820-1.

[24] H. Alam, Lagrangian relaxation method for multiobjective optimization methods: Solution approaches., Journal of Applied Mathematics and Physics 10 (2022) 1619–1630. doi:https://doi.org/10.4236/jamp.2022.105112.

[25] A. Dunbar, S. Sinha, A. J. Schaefer, Relaxations and duality for multiobjective integer programming, Math. Program. 207 (1–2) (2023) 577–616. doi:10.1007/s10107-023-02022-7.

[26] M. Brun, T. Perini, S. Sinha, A. J. Schaefer, On the strength of lagrangian duality in multiobjective integer programming, Mathematical Programming 212 (1) (2025) 683–715. doi:10.1007/s10107-024-02121-z.

[27] J. Bauß, S. N. Parragh, M. Stiglmayr, On improvements of multi-objective branch and bound, EURO Journal on Computational Optimization 12 (2024) 100099. doi:https://doi.org/10.1016/j.ejco.2024.100099.
URL https://www.sciencedirect.com/science/article/pii/S2192440624000169

[28] L. A. Wolsey, Integer programming, John Wiley & Sons, Hoboken, NJ, 2020. doi:https://doi.org/10.1002/9781119606475.oth1.

[29] N. Z. Shor, K. C. Kiwiel, A. Ruszcayundefinedski, Minimization methods for non-differentiable functions, Springer-Verlag, Berlin, Heidelberg, 1985.

[30] J.-L. Goffin, On convergence rates of subgradient optimization methods, Mathematical programming 13 (1) (1977) 329–347.

[31] M. L. Fisher, The lagrangian relaxation method for solving integer programming problems, Management Science 27 (1) (1981) 1–18. doi:https://doi.org/10.1287/mnsc.27.1.1.

[32] M. L. Fisher, An applications oriented guide to lagrangian relaxation, Interfaces 15 (2) (1985) 10–21. doi:https://doi.org/10.1287/inte.15.2.10.

[33] M. Held, P. Wolfe, H. P. Crowder, Validation of subgradient optimization, Math. Program. 6 (1) (1974) 62–88. doi:10.1007/BF01580223.
URL https://doi.org/10.1007/BF01580223

[34] B. T. Polyak, Gradient methods for the minimisation of functionals, USSR Computational Mathematics and Mathematical Physics 3 (4) (1963) 864–878.

[35] N. Forget, S. L. Gadegaard, K. Klamroth, L. R. Nielsen, A. Przybylski, Branch-and-bound and objective branching with three or more objectives, Computers & Operations Research 148 (2022) 106012. doi:https://doi.org/10.1016/j.cor.2022.106012.
URL https://www.sciencedirect.com/science/article/pii/S030505482200243X

[36] Y. P. Aneja, K. P. K. Nair, Bicriteria transportation problem, Management Science 25 (1) (1979) 73–78. doi:https://doi.org/10.1287/mnsc.25.1.73.

[37] E. Fernández, J. Puerto, Multiobjective solution of the uncapacitated plant location problem, European Journal of Operational Research 145 (3) (2003) 509–529. doi:https://doi.org/10.1016/S0377-2217(02)00223-0.
URL https://www.sciencedirect.com/science/article/pii/S0377221702002230

[38] T. Bektaş, Disjunctive programming for multiobjective discrete optimisation, INFORMS Journal on Computing 30 (4) (2018) 625–633. doi:10.1287/ijoc.2018.0812.