Chapter 5
Multi-Objective Lexicographic Mixed-Integer Linear Programming: an Infinity Computer Approach

Marco Cococcioni, Alessandro Cudazzo, Massimo Pappalardo, Yaroslav D. Sergeyev

Abstract In this chapter we show how a lexicographic multi-objective linear programming problem (LMOLP) can be transformed into an equivalent, single-objective one, by using the Grossone Methodology. Then we provide a simplex-like algorithm, called GrossSimplex, able to solve the original LMOLP problem using a single run of the algorithm (its theoretical correctness is also provided). In the second part, we tackle a Mixed-Integer Lexicographic Multi-Objective Linear Programming problem (LMOMILP) and we solve it in an exact way, by using a Grossone-version of the Branch-and-Bound scheme (called GrossBB). After proving the theoretical correctness of the associated pruning rules and terminating conditions, we show a few experimental results, run on an Infinity Computer simulator.

5.1 Introduction

In this contribution we survey recent achievements in the field of lexicographic linear programming by providing a coherent mathematical framework for the main results obtained in [6] and [2].

Marco Cococcioni
Dipartimento di Ingegneria dell'Informazione (University of Pisa), Largo Lucio Lazzarino, 1 – 56122 Pisa, Italy e-mail: marco.cococcioni@unipi.it

Massimo Pappalardo, Alessandro Cudazzo
Dipartimento di Informatica (University of Pisa), Largo B. Pontecorvo, 3 – 56127 Pisa, Italy e-mail: massimo.pappalardo@unipi.it,a.cudazzo1@studenti.unipi.it

Yaroslav D. Sergeyev
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (University of Calabria), via P. Bucci, Cubo 41-C – 87036 Rende (CS), Italy, and
Lobachevsky State University, Nizhni Novgorod, Russia
e-mail: yaro@dimes.unical.it

Lexicographic multi-objective optimization problem consists of finding the solution that optimizes the first (most important) objective and, only if there are multiple equally-optimal solutions, find the one that optimizes the second most important objectives, and so on.

Lexicographic Multi-Objective Linear Programming (LMOLP) consists of the solution of an optimization problem having multiple linear objective function, sorted in order of priority (*lexicographic property*), over a region defined by linear constraints. If the search region is closed, bounded and not empty, then at least one solution exists. Sometimes the solution is also unique, despite the fact the problem is multi-objective. This is due to the lexicographic property. Indeed, while a Pareto Multi-Objective LP problem in general admits multiple Pareto optimal solutions (non-dominated solutions), the lexicographic attribute tends to reduce the number of solutions to a single optimal one.

In this book chapter we present a technique, based on Grossone [26], to transform an LMOLP problem into a single objective one (this technique is known as *scalarization*). Then we present a generalization of the simplex algorithm, that we have called GrossSimplex algorithm, which is able to solve the Grossone-based and scalarized single-objective LP problem. After showing an illustrative LMOLP problem, solved by the GrossSimplex algorithm, we move to introduce the LMOMILP problem. Then we show how to solve it by using a Grossone-based extension of the well-known Branch-and-Bound method: we called this algorithm *GrossBB*. Finally we present some test problems having a known solution, and then we demonstrate that the GrossBB algorithm, coupled with the GrossSimplex, is able to correctly solve all the considered problems. Before continuing, it is important to remind that the Grossone Methodology is independent from non-standard analysis, as discussed in [27].

Concerning related works, it must be acknowledged that the first successful attempt to use Grossone Methodology within LP problems is due to [7]. The same author together with his co-authors has proposed other applications of Grossone to optimization problems, in [8] and [9, 10]. Concerning global optimization, Grossone has been used in [28], while regarding convex non-smooth optimization it has been used in [13]. As regard as evolutionary optimization, four works have already been presented [15–18]. Grossone Methodology has also been applied to Game Theory [4, 11, 12], infinite decision processes science [21–23], ordinary differential equations [1, 24, 29], among other fields.

The chapter is organized as follows. In next Section we introduce the LMOLP problem, while in Section 5.3 we show its transformation into a single objective problem using the Grossone Methodology. Within the same Section, we prove that the original formulation and the Grossone-based one are equivalent. Then in Section 5.4 we introduce the GrossSimplex algorithm (an extension to the simplex algorithm able to work with Grossone-based numbers) and we show an example of its execution. In Section 5.5 we introduce the LMOMILP problem, and its reformulation using Grossone, again proving their equivalence. In Section 5.6 we provide a Grossone-based extension to a classical Branch-and-Bound algorithm. Section 5.7 is devoted to the experimental results: it shows how the GrossBB algorithm (coupled with the

GrossSimplex one) is able to solve three LMOMILP problems. Finally, Section 5.8 provides a few conclusions.

5.2 Lexicographic multi-objective linear programming

Given the LMOLP problem:

$$\begin{aligned} \text{LexMax} \quad & \mathbf{c}^{1T}\mathbf{x},\ \mathbf{c}^{2T}\mathbf{x},\ ...,\ \mathbf{c}^{rT}\mathbf{x} \\ \text{s.t.} \quad & \{\mathbf{x}\in\mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqslant 0\} \end{aligned} \qquad \text{P1}$$

where $\mathbf{x}$ and $\mathbf{c}^i,\ i = 1, ..., r$, are column vectors$\in\mathbb{R}^n$, $\mathbf{A}$ is a full-rank matrix$\in\mathbb{R}^{m\times n}$, $\mathbf{b}$ is a column vector$\in\mathbb{R}^m$. LexMax in P1 denotes *lexicographic maximum* and means that the first objective is much more important than the second, which is, on its turn, much more important than the third one, and so on. Sometimes in the literature this is denoted as $\mathbf{c}^{1T}\mathbf{x} \gg \mathbf{c}^{2T}\mathbf{x} \gg ... \gg \mathbf{c}^{rT}\mathbf{x}$.
As in any LP problem, the domain of P1 is a polyhedron:

$$\mathcal{S} \equiv \{x\in\mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqslant 0\}\,. \tag{5.1}$$

Notice that the formulation of P1 makes no use of Gross-numbers or Gross-arrays involving ①, namely, it involves finite numbers only. Hereinafter we assume that $\mathcal{S}$ is bounded and non-empty. In the literature (see, e.g., [14, 20, 30, 31]), there exists two approaches for solving the problem P1: the *preemptive scheme* and the *nonpreemptive scheme*. They are described in the following two subsections.

5.2.1 The preemptive scheme

The preemptive scheme introduced in [30] is an iterative method that attacks P1 by solving a series of single-objective LP problems. It starts by considering the first objective function alone, i.e., by solving the following problem:

$$\begin{aligned} \text{Max} \quad & \mathbf{c}^{1T}\mathbf{x} \\ \text{s.t.} \quad & \{\mathbf{x}\in\mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqslant 0\} \end{aligned} \qquad \text{P2.1}$$

Since P2.1 is a canonical LP problem, it can be solved algorithmically using any standard method (e.g., the simplex algorithm, the interior point algorithm, etc.). Once they have been run, an optimal solution $\mathbf{x}^{*1}$ with the optimal value $\beta_1 = \mathbf{c}^{1T}\mathbf{x}^{*1}$ has been obtained. Then the preemptive scheme considers the second objective of P1 and solves another single-objective LP problem, where the domain has changed, due to the addition of the equality constraint $\mathbf{c}^{1T}\mathbf{x} = \beta_1$:

$$\begin{aligned} \text{Max} \quad & \mathbf{c}^{2T}\mathbf{x} \\ \text{s.t.} \quad & \left\{\mathbf{x}\in\mathbb{R}^n : \mathbf{A}\mathbf{x}=\mathbf{b},\ \mathbf{x}\geqslant 0,\ \mathbf{c}^{1T}\mathbf{x}=\beta_1\right\} \end{aligned} \tag{P2.2}$$

The same approach is reiterated, and the algorithm stops either when the last problem has been solved, i.e., after considering the last objective $\mathbf{c}^{rT}\mathbf{x}$ or when a unique solution has been found in the current solved LP problem. Clearly, this approach is time consuming.

5.2.2 The nonpreemptive scheme based on appropriate finite weights

It has been shown in [30] that there always exists a finite scalar $M\in\mathbb{R}$ such that the solution of the LMOLP problem P1 can be found by solving only one single-objective LP problem having the following form:

$$\begin{aligned} \text{Max} \quad & \bar{\mathbf{c}}^{T}\mathbf{x} \\ \text{s.t.} \quad & \{\mathbf{x}\in\mathbb{R}^n : \mathbf{A}\mathbf{x}=\mathbf{b},\ \mathbf{x}\geqslant 0\} \end{aligned} \tag{P3}$$

where

$$\bar{\mathbf{c}} = \sum_{i=1}^{r} \mathbf{c}^{i} M^{-i+1}. \tag{5.2}$$

This is a powerful theoretical result. However, from the computational point of view, finding the value of M is not a trivial task. Finding an appropriate value of M and solving the resulting LP problem can be more time consuming than solving the original problem P1 following the preemptive approach. Indeed, the preemptive scheme requires solving r linear programming problems only in the worst case and, in addition, it does not require the computation of M.

In Section 5.3, we present a nonpreemptive approach based on infinitesimal weights (constructed by using Grossone integer powers), that overcomes the problem of computing M and still requires the solution of only one single-objective LP problem. Such an LP problem is, however, not a standard one and thus it will be called Gross-LP problem, to avoid any possible confusion with its standard formulation involving finite numbers only.

5.3 Grossone-based reformulation of the LMOLP problem

In this Section we will introduce a nonpreemptive Grossone-based scheme for addressing LMOLP problems. In order to introduce such scheme we need the following definitions and notations. Hereinafter, an array made of Gross-scalars is called Gross-array. In particular, a vector made of Gross-scalars is called from here on Gross-vector. The definition of Gross-matrix is similar. By extension, Gross-vectors

Table 5.1 Notation used in this work

Font style	Example	Quantity
Italics lowercase	n	purely finite real scalar
Boldface lowercase	$\mathbf{x}$	purely finite real vector
Boldface uppercase	$\mathbf{A}$	purely finite real matrix
Italics lowercase with tilde	$\tilde{c}$	Gross-scalar
Boldface lowercase with tilde	$\tilde{\mathbf{y}}$	Gross-vector

and Gross-matrices are called *purely finite* iff all of their entries are purely finite Gross-numbers (where we have defined a purely finite Gross-number as a Gross-number involving a single power of Grossone, and that power has zero as its exponent: examples are $3.56①^0$, 0.479, etc.). The used notation is summarized in Tab. 5.1.

Let us introduce now the nonpreemptive Grossone-based scheme following the lexicographic ①-based approach introduced in [5, 6, 25]. It should be stressed that it is supposed hereinafter that the original problem P1 has been stated using purely finite numbers only. This assumption is not restrictive from the practical point of view since all the LMOLP problems considered traditionally are of this kind. However, since we are now in the framework of ①-based numbers, this assumption should be explicitly stated.

To state the problem P4, we reformulate P3 by making use of Gross-scalars and Gross-vectors and is defined as follows:

$$\begin{aligned} \text{Max} \quad & \tilde{\mathbf{c}}^T \mathbf{x} \\ \text{s.t.} \quad & \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqslant 0\}, \end{aligned} \tag{P4}$$

where $\tilde{\mathbf{c}}$ is a column Gross-vector having n Gross-scalar components:

$$\tilde{\mathbf{c}} = \sum_{i=1}^{r} \mathbf{c}^i ①^{-i+1} \tag{5.3}$$

and $\tilde{\mathbf{c}}^T \mathbf{x}$ is the Gross-scalar obtained by multiplying the Gross-vector $\tilde{\mathbf{c}}$ by purely finite vector $\mathbf{x}$

$$\tilde{\mathbf{c}}^T \mathbf{x} = (\mathbf{c}^{1T} \mathbf{x})①^0 + (\mathbf{c}^{2T} \mathbf{x})①^{-1} + \ldots + (\mathbf{c}^{rT} \mathbf{x})①^{-r+1}, \tag{5.4}$$

where (5.4) can be equivalently written in the extended form as

$$\tilde{\mathbf{c}}^T \mathbf{x} = (c_1^1 x_1 + \ldots + c_n^1 x_n)①^0 + (c_1^2 x_1 + \ldots + c_n^2 x_n)①^{-1} + \ldots + (c_1^r x_1 + \ldots + c_n^r x_n)①^{-r+1}.$$

The fundamental difference between the definition of $\tilde{\mathbf{c}}$ in (5.3) with respect to the definition of $\bar{\mathbf{c}}$ in (5.2) is that $\tilde{\mathbf{c}}$ does not involve any unknown. More precisely, it does not require the specification of a real scalar value, like the value of M in P3. However, this advantage leads to the fact that standard algorithms (like the simplex algorithm or the interior point algorithm), traditionally used for solving P3, can no longer be used in this case since ①-based numbers are involved in the definition of the objective

function. There will be shown in the next section that it is still possible to obtain optimality conditions and to introduce and implement a GrossSimplex algorithm (a generalization of the traditional simplex algorithm to the case of ①-based numbers) able to solve problem P4.

In the rest of this section we will prove that problems P1 and P4 are equivalent and then we will derive the optimality conditions. Before giving the equivalence theorem, the following three lemmas are required.

The first lemma states that all the optimal solutions of P4 are vertices or belong to the convex hull of vertices, where the set of all the optimal solutions for a generic problem P is denoted, from hereafter, by the symbol Ω(P).

Lemma 1 *Each* $\mathbf{x}^* \in \Omega(\text{P4})$ *is a vertex or lies on the convex hull of the optimal vertices.*

The proof of Lemma 1 can be found in the appendix.

The next lemma states that all the optimal solutions of P1 reach the same (Gross-scalar) objective value for problem P4.

Lemma 2 *For any* $\mathbf{x}^* \in \Omega(\text{P1})$ *it follows* $\tilde{\mathbf{c}}^T \mathbf{x}^* = \tilde{v}$ ($\tilde{v}$ being a constant Gross-scalar).

Again, the proof of Lemma 2 is given in the appendix.

The third lemma is the last step in preparing the proof of the equivalence between P1 and P4.

Lemma 3 *For all* $\mathbf{x}^* \in \Omega(\text{P1})$ *and any vertex* $\hat{\mathbf{x}}$ *of S such that* $\hat{\mathbf{x}} \notin \Omega(\text{P1})$, *it follows*

$$\tilde{\mathbf{c}}^T \hat{\mathbf{x}} \; < \; \tilde{\mathbf{c}}^T \mathbf{x}^*.$$

As before, the proof is reported in the appendix.

We are now ready to prove that any solution to P1 is also a solution to P4, and vice-versa, as shown in next theorem.

Theorem 1 (Equivalence) $\mathbf{x}^* \in \Omega(\text{P1})$ *iff* $\mathbf{x}^* \in \Omega(\text{P4})$.

Proof $\Rightarrow$ If $\mathbf{x}^*$ is optimal for P1, then $\tilde{\mathbf{c}}^T \mathbf{x}^* = \tilde{v}$ due to lemma 2. Therefore, according to lemma 1 and 3, $\mathbf{x}^*$ is also optimal for P4.

$\Leftarrow$ If $\mathbf{x}^*$ is optimal for P4, then due to lemma 1, it belongs to the convex hull of some vertices. But from lemma 3 it follows that all the vertices of this kind are optimal solutions to problem P1. □

5.4 The GrossSimplex algorithm

The single-objective Gross-LP problem formulated in P4 using Grossone can be solved using the GrossSimplex algorithm here described, provided that the duality theory is extended to the case of Gross-scalars and Gross-vectors.

The dual problem of P4 is the following

$$\begin{aligned} \text{Min} \quad & \tilde{\mathbf{y}}^T \mathbf{b} \\ \text{s.t.} \quad & \left\{ \tilde{\mathbf{y}} : \ \mathbf{A}^T \tilde{\mathbf{y}} \geqslant \tilde{\mathbf{c}} \right\}, \end{aligned} \tag{P5}$$

where $\tilde{\mathbf{y}} = [\tilde{y}_1, ..., \tilde{y}_m]^T$ is an m-dimensional column Gross-vector, $\mathbf{A}^T \tilde{\mathbf{y}}$ an n-dimensional column Gross-vector, and $\tilde{\mathbf{y}}^T \mathbf{b}$ a Gross-scalar.
The domain of P5 is the Gross-polyhedron $\tilde{\mathcal{D}}$ defined as:

$$\tilde{\mathcal{D}} \equiv \left\{ \tilde{\mathbf{y}} : \ \mathbf{A}^T \tilde{\mathbf{y}} \geqslant \tilde{\mathbf{c}} \right\}.$$

In the following we will assume that $\tilde{\mathcal{D}}$ is bounded and non-empty, as we have assumed for $\mathcal{S}$.

Since optimality conditions are the core of the simplex algorithm, we need to prove optimality conditions in our context.

Definition 1 (Definition of optimality for the Gross-primal problem) A point $\mathbf{x}^*$ is optimal for the problem P4 iff $\tilde{\mathbf{c}}^T \mathbf{x}^* \geqslant \tilde{\mathbf{c}}^T \mathbf{x} \quad \forall \mathbf{x} \in \mathcal{S}$.

Definition 2 (Definition of optimality for the Gross-dual problem) A point $\tilde{\mathbf{y}}^*$ is optimal for the problem P5 iff $\tilde{\mathbf{y}}^{*T} \mathbf{b} \leqslant \tilde{\mathbf{y}}^T \mathbf{b} \quad \forall \tilde{\mathbf{y}} \in \tilde{\mathcal{D}}$.

We need now the following lemma.

Lemma 4 (weak duality) $\forall \mathbf{x} \in \mathcal{S}$ *and* $\forall \tilde{\mathbf{y}} \in \tilde{\mathcal{D}}$, *it follows that* $\tilde{\mathbf{y}}^T \mathbf{b} \geqslant \tilde{\mathbf{c}}^T \mathbf{x}$.

Proof Since $\mathbf{x} \in \mathcal{S}$, $\mathbf{A}\mathbf{x} = \mathbf{b}$. Pre-multiplying both by $\tilde{\mathbf{y}}^T$, we have

$$\tilde{\mathbf{y}}^T \mathbf{A}\mathbf{x} = \tilde{\mathbf{y}}^T \mathbf{b}. \tag{5.5}$$

Now, since $\tilde{\mathbf{y}} \in \tilde{\mathcal{D}}$, we know that $\tilde{\mathbf{y}}^T \mathbf{A} \geqslant \tilde{\mathbf{c}}^T$. By post-multiplying the latter by $\mathbf{x}$ (being $\mathbf{x} \geqslant 0$), we have:

$$\tilde{\mathbf{y}}^T \mathbf{A}\mathbf{x} \geqslant \tilde{\mathbf{c}}^T \mathbf{x}.$$

By combining it with (5.5) we obtain $\tilde{\mathbf{y}}^T \mathbf{b} \geqslant \tilde{\mathbf{c}}^T \mathbf{x}$. □

Theorem 2 (Optimality condition) *If* $\mathbf{x}^* \in \mathcal{S}$, $\tilde{\mathbf{y}}^* \in \tilde{D}$ *and* $\tilde{\mathbf{c}}^T \mathbf{x}^* = \tilde{\mathbf{y}}^{*T} \mathbf{b}$, *then it follows that* $\mathbf{x}^* \in \Omega$(P1) *and* $\tilde{\mathbf{y}}^* \in \Omega$(P5).

Proof It is an immediate consequence of the weak duality lemma 4. □

We are now ready to introduce the GrossSimplex algorithm (see Algorithm 1) which exploits the theoretical results presented above. Notice that the algorithm needs a first feasible basis B. It can be found by solving the standard auxiliary problem

Algorithm 1 The GrossSimplex algorithm

Step 0. The user has to provide the initial set $\mathtt{B}$ of basic indices.

Step 1. Solve the system $\mathbf{A}_{\mathtt{B}}^T \tilde{\mathbf{y}} = \tilde{\mathbf{c}}_{\mathtt{B}}$ (where $\mathbf{A}_{\mathtt{B}}^T$ is the sub-matrix obtained by $\mathbf{A}^T$ by considering the columns indexed by $\mathtt{B}$).

This can be easily calculated as $\tilde{\mathbf{y}} = \mathbf{A}_{\mathtt{B}}^{-T} \tilde{\mathbf{c}}_{\mathtt{B}}$ where $\mathbf{A}_{\mathtt{B}}^{-T}$ is an abbreviation for $\left(\mathbf{A}_{\mathtt{B}}^T\right)^{-1}$, $\tilde{\mathbf{y}}$ is a Gross-vector obtained by linearly combining the Gross-vector $\tilde{\mathbf{c}}_{\mathtt{B}}$ using the purely finite scalar elements in $\mathbf{A}_{\mathtt{B}}^{-T}$.

Step 2. Compute $\tilde{\mathbf{s}} = \tilde{\mathbf{c}}_{\mathtt{N}} - A_{\mathtt{N}}^T \tilde{\mathbf{y}}$ (where $\mathtt{N}$ is the complementary set of $\mathtt{B}$) and then select the maximum (gradient rule). When this maximum is negative (being it finite or infinitesimal), it means that we are done (the current solution is optimal) and thus the algorithm STOPS. Otherwise, the position of the maximum in the Gross-vector $\tilde{\mathbf{s}}$ is the index k of the entering variable $\mathtt{N}(k)$.

Step 3. Solve the system $\mathbf{A}_{\mathtt{B}}\mathbf{d} = \mathbf{A}_{\mathtt{N}(k)}$.

Step 4. Find the largest t such that $\mathbf{x}_{\mathtt{B}}^* - t\mathbf{d} \geqslant 0$. If there is not such a t, then the problem is unbounded (STOP); otherwise, at least one component of $\mathbf{x}_{\mathtt{B}}^* - t\mathbf{d}$ equals zero and the corresponding variable is the leaving variable. In case of ties, use the Lexicographic Pivoting Rule [7] to break them.

Step 5. Update $\mathtt{B}$ and $\mathtt{N}$ and return to Step 1.

analogously to the case where the objective function is only one (no GrossSimplex required to solve it).

To illustrate the behaviour of the GrossSimplex algorithm we consider the following problem:

$$\begin{aligned} \text{LexMax} \quad & 4x_1 + 2x_2, \quad 2x_2, \quad x_1 + x_2 \\ \text{s.t.} \quad & 2x_1 \quad + x_2 \quad \leq 14 \\ & -x_1 + 2x_2 \leq 8 \\ & 2x_1 \quad - x_2 \leq 10 \\ & x_1, x_2 \geq 0 \end{aligned}$$

The polygon $\mathcal{S}$ associated to this problem is shown in Fig. 5.1. It can be seen that the first objective vector $\mathbf{c}^1 = [4, 2]^T$ is orthogonal to segment $[\alpha, \beta]$ ($\alpha = (6, 2)$, $\beta = (4, 6)$) shown in the same figure. Thus all the points laying on this segment are optimal. Since the solution is not unique, there is the chance to try to improve the second objective vector ($\mathbf{c}^2 = [0, 2]^T$) without deteriorating the first one function. The point that maximizes the second objective is β, associated to solution $\mathbf{x}^* = [4, 6]^T$. Since now the solution is unique, the third objective function can not be taken into account. Thus the lexicographic optimal solution for the problem is $\mathbf{x}^* = [4, 6]^T$.

Before running the GrossSimplex algorithm we had to transform the problem into the following one, after converting the constraints into equality constraints by adding slack variables x_3, x_4, and x_5:

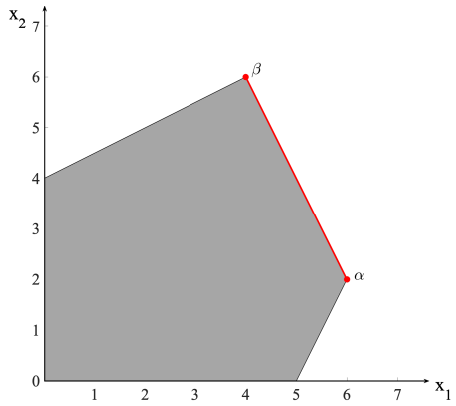

Fig. 5.1 An example in two dimensions with three objectives. All the points in the segment $[\alpha, \beta]$ are optimal for the first objective, while point β is the unique lexicographic optimum for the given problem ($\beta = (4, 6)$).

$$
\begin{array}{rl}
\text{LexMax} & 4x_1 + 2x_2, \quad 2x_2, \quad x_1 + x_2 \\
\text{s.t.} & 2x_1 \quad + x_2 \quad + x_3 = 14 \\
& -x_1 + 2x_2 + x_4 = 8 \\
& 2x_1 \quad - x_2 \quad + x_5 = 10 \\
& x_i \geq 0 \quad i = 1, ..., 5
\end{array} \qquad T_0
$$

The GrossSimplex algorithm has been run on problem T_0, by using the initial basis $\mathtt{B} = \{3, 4, 5\}$ ($\mathtt{N}$ is therefore $\{1, 2\}$). The initial solution associated to the initial basis is: $\mathbf{x} = [0, 0, 14, 8, 10]^T$, which corresponds to the point (0, 0) in Fig. 5.1. Then the algorithm computes $\tilde{\mathbf{s}}$ as $\tilde{\mathbf{c}}_\mathtt{N} - A_\mathtt{N}^T \tilde{\mathbf{y}}$ giving

$$
\tilde{\mathbf{s}} = \begin{bmatrix} 4①^0 + 0①^{-1} + 1①^{-2} \\ 2①^0 + 2①^{-1} + 1①^{-2} \end{bmatrix}.
$$

Thus, according to the gradient rule for choosing the entering variable, the one in $\mathtt{N}$ having the first index is selected, i.e., x_1, which is the one associated to the maximum constraint violation: $4①^0 + 0①^{-1} + 1①^{-2}$. The leaving index, computed according to the lexicographic pivoting rule is the third, i.e., variable x_5. Thus the second base used is $\mathtt{B} = \{3, 4, 1\}$. The Gross-objective function is $\tilde{\mathbf{c}}^T \mathbf{x} = 20.00①^0 + 0.00①^{-1} + 5.00①^{-2}$ and, of course, coincides with that of the dual $\tilde{\mathbf{y}}^T \mathbf{b}$ (when $\tilde{\mathbf{y}}$ will be also feasible, we will be at the optimum and the algorithm will end).

The solution associated to this second base is $\mathbf{x} = [5, 0, 4, 13, 0]^T$, while the new value for $\tilde{\mathbf{s}}$ is

$$
\tilde{\mathbf{s}} = \begin{bmatrix} -2①^0 + 0①^{-1} - 0.5①^{-2} \\ 4①^0 + 2①^{-1} + 1.5①^{-2} \end{bmatrix}.
$$

Again, according to the gradient rule, the next entering index is the second, i.e., variable x_2. The leaving index is the first and thus the leaving variable is x_3. Therefore, the next basis (the third) is $\mathtt{B} = \{2, 4, 1\}$. The Gross-objective function is equal to $28.00①^0 + 4.00①^{-1} + 8.00①^{-2}$.

The solution associated to the third base is $\mathbf{x} = [6, 2, 0, 10, 0]^T$, while the new value for $\tilde{\mathbf{s}}$ is

$$\tilde{\mathbf{s}} = \begin{bmatrix} 0①^0 + 1①^{-1} + 0.25①^{-2} \\ -2①^0 - 1①^{-1} - 0.75①^{-2} \end{bmatrix}.$$

By following the same process, the next entering index is the first, i.e., variable x_5. It is interesting to note that in this case the constraint violation (a positive entry in $\tilde{\mathbf{s}}$) is not finite, as in previous case, but infinitesimal: $+1①^{-1} + 0.25①^{-2}$ The leaving index is the second and thus the leaving variable is x_4.

At this point, the Gross-objective function is now equal to $28.00①^0 + 12.00①^{-1} + 10.00①^{-2}$, the solution associated to the fourth base is $\mathbf{x} = [4, 6, 0, 0, 8]^T$, and the $\tilde{\mathbf{s}}$ is

$$\tilde{\mathbf{s}} = \begin{bmatrix} 0①^0 - 0.8①^{-1} - 0.2①^{-2} \\ -2①^0 - 0.4①^{-1} - 0.6①^{-2} \end{bmatrix}.$$

Since all the entries of $\tilde{\mathbf{s}}$ are negative (one is infinitesimal and one is finite), we are done. In fact, the solution found (once the slack variables are discarded), is the correct one, i.e., $\mathbf{x}^* = [4, 6]^T$. Furthermore, the Gross-objective function at the end equals to $\tilde{\mathbf{c}}^T\mathbf{x} = \tilde{\mathbf{y}}^T\mathbf{b} = 28.00①^0 + 12.00①^{-1} + 10.00①^{-2}$. This concludes the step-by-step illustration about how the GrossSimplex works in a concrete example.

As a final note, observe how in [3] a parameter-less way to obtain the initial basis is shown, based on the Big-M method with an infinitely big value for M. The idea of that method is to use a numerical, infinitely big value for M, set equal to ①.

In the next section we present the Lexicographic Multi-Objective Mixed-Integer Linear Programming problem and then, in the subsequent version, how to solve it using a Grossone-based extension of the Branch-and-Bound method.

5.5 Lexicographic Multi-Objective Mixed-Integer Linear Programming (LMOMILP)

In this section we introduce the LMOMILP problem, which is formalized as:

$$P \quad \begin{array}{ll} \text{LexMin} & \mathbf{c}^{1T}\mathbf{x},\ \mathbf{c}^{2T}\mathbf{x}, ..., \mathbf{c}^{rT}\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \\ & \mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \qquad \mathbf{p} \in \mathbb{Z}^k, \quad \mathbf{q} \in \mathbb{R}^{n-k} \end{array}$$

where $\mathbf{c}^i,\ i = 1, ..., r$, are column vectors $\in \mathbb{R}^n$, $\mathbf{x}$ is a column vector $\in \mathbb{R}^n$, $\mathbf{A}$ is a full-rank matrix $\in \mathbb{R}^{m\times n}$, $\mathbf{b}$ is a column vector $\in \mathbb{R}^m$. LexMin in P denotes the lexicographic minimum.

As in any MILP problem, from the problem P, we can define the polyhedron defined by the linear constraints alone:

$$\mathcal{S} \equiv \{\mathbf{x}\in\mathbb{R}^n : \mathbf{A}\mathbf{x} \leqslant \mathbf{b}\}\,. \tag{5.6}$$

Thus we can define problem R, the *relaxation* of a lexicographic (mixed) integer linear problem, obtained from P by removing the integrality constraint on each variable:

$$\begin{array}{ll} \text{LexMin} & \mathbf{c}^{1\,T}\mathbf{x},\ \mathbf{c}^{2\,T}\mathbf{x}, ..., \mathbf{c}^{r\,T}\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \end{array} \qquad R$$

Problem R is called LMOLP (Lexicographic Multi-Objective Linear Problem), and can be solved using the GrossSimplex algorithm shown in Section 5.4. In addition, every thing we have said above for the MILP problem is still valid for the LMOMILP.

Notice that the formulation of P makes no use of Gross-numbers or Gross-arrays involving ①, namely, it involves finite numbers only. Hereinafter we assume that $\mathcal{S}$ is bounded and non-empty. In the next we will reformulate the LMOMILP problem P using Grossone, and then we will prove their equivalence.

We present now an equivalent reformulation of the LMOMILP problem P, based on Grossone Methodology.

First of all, let us introduce the new problem $\tilde{P}$, formulated using Gross-numbers, following the same approach shown in Section 5.3:

$$\begin{array}{lll} \text{Min} & \tilde{\mathbf{c}}^T\mathbf{x} & \\ \text{s.t.} & \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, & \\ & \mathbf{x} = \begin{bmatrix}\mathbf{p}\\ \mathbf{q}\end{bmatrix} \qquad \mathbf{p}\in\mathbb{Z}^k, & \mathbf{q}\in\mathbb{R}^{n-k}, \end{array} \qquad \tilde{P}$$

where $\tilde{\mathbf{c}}$ is a column Gross-vector having n Gross-scalar components built using purely finite vectors $\mathbf{c}^i$

$$\tilde{\mathbf{c}} = \sum_{i=1}^{r} \mathbf{c}^i ①^{-i+1} \tag{5.7}$$

and $\tilde{\mathbf{c}}^{\mathbf{T}}\mathbf{x}$ is the Gross-scalar obtained by multiplying the Gross-vector $\tilde{\mathbf{c}}$ by the purely finite vector $\mathbf{x}$

$$\tilde{\mathbf{c}}^{\mathbf{T}}\mathbf{x} = (\mathbf{c}^{1T}\mathbf{x})①^0 + (\mathbf{c}^{2T}\mathbf{x})①^{-1} + ... + (\mathbf{c}^{rT}\mathbf{x})①^{-r+1}. \tag{5.8}$$

Observe how equation (5.7) is identical to equation (5.3), and equation (5.8) is the same as equation (5.4).

In Theorem 2 we will prove that problem $\tilde{P}$ is equivalent to problem P defined on Section 5.5.

What makes the new formulation $\tilde{P}$ attractive is the fact that its relaxed version (from the integrality constraint) is a Gross-LP problem, and therefore it can be effectively solved using *a single run* of the GrossSimplex algorithm introduced in Section 5.4. This means that the set of multiple objective functions is mapped into a single (Gross-) scalar function to be optimized. This opens the possibility to solve the integer-constrained variant of the problem using an adaptation of the BB algorithm (see Algorithm 2), coupled with the GrossSimplex. Of course the GrossSimplex will solve the relaxed version of $\tilde{P}$:

$$\tilde{R} \quad \begin{aligned} \text{Min} \quad & \tilde{\mathbf{c}}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leqslant \mathbf{b} \end{aligned}$$

Theorem 3 (Equivalence of problem $\tilde{P}$ and problem P) *Problem $\tilde{P}$ is equivalent to the problem P. Thus the solutions of the two are the same.*

Proof The basic observation is that the integer relaxation R of problem P is an LMOLP problem, while the integer relaxation of problem $\tilde{P}$, the $\tilde{R}$ problem defined above, is a Gross-LP problem. In Section 5.3 we have already proved the equivalence of problems R and $\tilde{R}$. Now, since problems P and $\tilde{P}$ have equivalent relaxations, the two will also share the same solutions when the same integrality constraints will be taken into account on both. □

In the next subsections we will provide the pruning rules, terminating conditions and the branching rule, and then we will introduce the GrossBB algorithm, a generalization of the BB algorithm able to work with Gross-numbers.

5.6 A Grossone-based extension of the Branch-and-Bound algorithm

In this Section we introduce a Grossone-based extension of the Branch-and-Bound algorithm, called GrossBB, to solve the Grossone-based formulation of any LMOMILP problem. As in any Branch-and-Bound algorithm, we have to provide the pruning rules (and the proof of their correctness), the termination conditions and the the branching rules.

5.6.1 Pruning rules for the GrossBB

The pruning rules of a standard Branch-and-Bound algorithm (see [19]) can be adapted to the case of a Grossone reformulated version of any LMOMILP problem:

Theorem 4 (Pruning rules for the GrossBB) *Let $\mathbf{x}_{opt}$ be the best solution found so far for $\tilde{P}$, and let $\tilde{v}_S(\tilde{P}) = \tilde{\mathbf{c}}^T \mathbf{x}_{opt}$ be the current upper bound. Considering the current node $(\tilde{P}_c)$ and the associated problem $\tilde{P}_c$:*

1. *If the feasible region of problem $\tilde{P}_c$ is empty the sub-tree with root $(\tilde{P}_c)$ has no feasible solutions having values lower than $\tilde{\mathbf{c}}^T \mathbf{x}_{opt}$. So we can prune this node.*
2. *If $\tilde{v}_I(\tilde{P}_c) \geqslant \tilde{v}_S(\tilde{P})$, then we can prune at node $(\tilde{P}_c)$, since the sub-tree with root $(\tilde{P}_c)$ cannot have feasible solutions having a value lower than $\tilde{v}_S(\tilde{P})$.*
3. *If $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$ and the optimal solution $\bar{\mathbf{x}}$ of the relaxed problem $\tilde{R}_c$ is feasible for $\tilde{P}$, then $\bar{\mathbf{x}}$ is a better candidate solution for $\tilde{P}$, and thus we can update $\mathbf{x}_{opt}$ $(\mathbf{x}_{opt} = \bar{\mathbf{x}})$ and the value of the upper bound $(\tilde{v}_S(\tilde{P}) = \tilde{v}_I(\tilde{P}_c))$. Finally, prune this node according to the second rule.*

The correctness of the above pruning rules is provided below.

Proof (Pruning Rule 1) If the feasible region of the current problem $\tilde{R}_c$ is empty, then also the one of $\tilde{P}_c$ is too, since the domain of $\tilde{P}_c$ has additional constrains (the integrality constraints). Furthermore, all the domain of the problems in the leaves of the sub-tree having root in $\tilde{P}_c$ will be empty as well, since the domains of the leaves have all additional constrains with respect to $\tilde{R}_c$. This proves the correctness of the first pruning rule. □

Now let us prove the second pruning rule.

Proof (Pruning Rule 2) Let us consider the generic leaf P_{leaf} of the sub-tree having root $\tilde{P}_c$. Let us indicate with $\tilde{v}(\tilde{P}_c)$ the optimal value of the current problem $\tilde{P}_c$ (the one with the integer constraints). Then the values at the leaves below $\tilde{P}_c$ must be greater than or equal to $\tilde{v}(\tilde{P}_c)$:

$$\tilde{v}(\tilde{P}_{leaf}) \geqslant \tilde{v}(\tilde{P}_c) \qquad \forall \text{ leaf in SubTree}(\tilde{P}_c)$$

In fact, the domain of $\tilde{P}_c$ includes the ones of all the $\tilde{P}_{leaf}$, being each problem $\tilde{P}_{leaf}$ obtained by enriching $\tilde{P}_c$ with additional constraints. On the other hand,

$$\tilde{v}(\tilde{P}_c) \geqslant \tilde{v}_I(\tilde{P}_c)$$

since $\tilde{v}_I(\tilde{P}_c)$ is obtained as the optimal solution of $\tilde{R}_c$, a problem having a domain which includes the one of $\tilde{P}_c$. Thus the following chain of inequalities always old:

$$\tilde{v}(\tilde{P}_{leaf}) \geqslant \tilde{v}(\tilde{P}_c) \geqslant \tilde{v}_I(\tilde{P}_c) \qquad \forall \text{ leaf in SubTree}(\tilde{P}_c)$$

Now, if $\tilde{v}_I(\tilde{P}_c) \geqslant \tilde{v}_S(\tilde{P})$, we can add an element to the chain:

$$\tilde{v}(\tilde{P}_{leaf}) \geqslant \tilde{v}(\tilde{P}_c) \geqslant \tilde{v}_I(\tilde{P}_c) \geqslant \tilde{v}_S(\tilde{P}) \qquad \forall \text{ leaf in SubTree}(\tilde{P}_c)$$

from which we can conclude that

$$\tilde{v}(\tilde{P}_{leaf}) \geqslant \tilde{v}_S(\tilde{P}) \qquad \forall \text{ leaf in SubTree}(\tilde{P}_c).$$

This means that all the leaves of the current node will contain solutions that are worse (or equivalent) than the current upper bound. Thus the sub-tree rooted in $\tilde{P}_c$

can to be pruned (i.e., not explicitly explored). This proves the correctness of the second pruning rule. □

Before proving the pruning rule 3, let us observe how the pruning rule above prevents from solving multi-modal problems, in the sense that with such pruning rule we are only able to find a single optimum, not all the solutions that might have the same cost function. In other words, the proposed pruning rule does not allow to solve multi-modal problems, because we are deciding not to explore the sub-tree at a given node that could contain solutions having the same current optimal objective function value. To solve multi-modal problems, that rule must be applied only when

$$\tilde{v}_I(\tilde{P}_c) > \tilde{v}_S(\tilde{P}) \tag{5.9}$$

Proof (Pruning Rule 3) If $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$ and $\bar{\mathbf{x}}$ is feasible for $\tilde{P}$, (i.e., if all the components of $\bar{\mathbf{x}}$ that must be integer are actually ϵ-integer), we have found a better estimate for the upper bound of $\tilde{P}$, and thus we can update it:

$$\tilde{v}_S(\tilde{P}) = \tilde{v}_I(\tilde{P}_c).$$

As a result, now $\tilde{v}_I(\tilde{P}_c) = \tilde{v}_S(\tilde{P})$, and thus:

$$\tilde{v}(\tilde{P}_{leaf}) \geqslant \tilde{v}(\tilde{P}_c) \geqslant \tilde{v}_I(\tilde{P}_c) = \tilde{v}_S(\tilde{P}) \qquad \forall\, leaf \text{ in SubTree}(\tilde{P}_c)$$

then again we have that the sub-tree having root $\tilde{P}_c$ cannot contain better solutions than $\bar{\mathbf{x}}$:

$$\tilde{v}(\tilde{P}_{leaf}) \geqslant \tilde{v}_S(\tilde{P}) \qquad \forall \text{ leaf in SubTree}(\tilde{P}_c).$$

This proves the correctness of the third pruning rule. □

5.6.2 Terminating conditions, branching rules and the GrossBB algorithm

Let us now discuss the terminating conditions for the GrossBB algorithm. The first two are exactly the same of the classical BB, while the third requires some attention.

The terminating conditions are:

1. **All the remaining leaves have been visited**: if all the leaves have been visited the GrossBB algorithm stops.
2. **Maximum number of iteration reached**: when a given maximum number of iterations (provided by the user at the beginning) has been reached, the GrossBB stops.
3. $\tilde{\epsilon}$**−optimality reached**: when the normalized difference between the global lower bound and the global upper bound at the i-th iteration is close enough to zero, we can stop:

$$\tilde{\Delta}^i(\tilde{P}) = \frac{\tilde{v}_S(\tilde{P}) - \tilde{v}_I(\tilde{P})}{|\tilde{v}_S(\tilde{P})|} \quad \leqslant \quad \tilde{\epsilon} \tag{5.10}$$

The last terminating condition deserves additional attention. It first involves the difference between two Gross-scalars. This intermediate result must be divided by the absolute value of $\tilde{v}_S(\tilde{P})$. While computing the absolute value of a Gross-scalar is straightforward, computing the division requires the algorithm described in [26]. The result of the Gross-division is a Gross-scalar that must be compared with the Gross-scalar $\tilde{\epsilon}$, which has the form:

$$\tilde{\epsilon} = \epsilon_0 + \epsilon_1①^{-1} + \epsilon_2①^{-2} + \ldots + \epsilon_{r-1}①^{-r+1}$$

Obviously, it is possible to chose $\epsilon_0 = \epsilon_1 = \epsilon_2 \ldots = \epsilon$, to simplify the presentation.

Let see the example below. Suppose to have three objectives (r=3) and to have selected $\epsilon = 10^{-6}$. Given the following $\tilde{\Delta}^i(\tilde{P})$

$$\tilde{\Delta}^i(\tilde{P}) = 1.1 \cdot 10^{-7} + 5 \cdot 10^{-3}①^{-1} + 1.7 \cdot 10^{-8}①^{-2}$$

it is clear that

$$\tilde{\Delta}^i(\tilde{P}) \not\leqslant \tilde{\epsilon}$$

because its first-order infinitesimal component ($5 \cdot 10^{-3}$) is not less or equal to ϵ. Thus in this case the GrossBB algorithm cannot terminate: it will continue, trying to make *all* the components $\leqslant \epsilon$.

Concerning the branching rule for the GrossBB algorithm, it works as follows. When the sub-tree below $\tilde{P}_c$ cannot be pruned (because it could contain better solutions), its sub-tree must be explored. Thus we have to branch the current node into P_l and P_r and to add these two new nodes to the tail of the queue of the sub-problems to be analyzed and solved by the GrossSimplex.

We are now ready to provide the pseudo-code for the GrossBB (see Algorithm 2). The GrossBB algorithm is able to solve a given $\tilde{P}$ LMOMILP problem, by internally using the GrossSimplex and the GrossBB rules provided above (pruning, terminating, and branching) .

This would allow us to create a division-free variant of the GrossBB algorithm. Anyway, we think that the use of the division is still interesting from the theoretical point of view, because it allowed us to clarify the concept of when a Gross-number is "near zero". Furthermore, the impact of the computation of the division of equation (5.10) on the overall computing time of the algorithm is negligible, being the overall computing time mainly affected by the time needed to compute the solutions of the relaxed LMOLP problems.

Algorithm 2 The GrossSimplex-based GrossBB Algorithm

Inputs: maxIter and a specific LMOMILP problem $|\tilde{P}|$, to be put within the root node $(\tilde{P})$
Outputs: $\mathbf{x}_{opt}$ (the optimal solution, a purely finite vector), $\tilde{f}_{opt}$ (the optimal value, a Gross-scalar)

Step 0. Insert $|\tilde{P}|$ into a queue of the sub problems that must be solved. Put $\tilde{v}_S(\tilde{P}) = ①$, $\mathbf{x}_{opt} = [\ ]$, and $\tilde{f}_{opt} = ①$ or use a greedy algorithm to get an initial feasible solution.

Step 1a. If all the remaining leaves have been visited (empty queue), or the maximum number of iterations has been reached, or the $\tilde{\epsilon}$-optimality condition holds, then goto Step 4. Otherwise extract from the head of the queue the next problem to solve and call it $\tilde{P}_c$ (*current problem*). Remark: this policy of insertion of new problems at the tail of the queue and the extraction from its head leads to a *breadth-first* visit for the binary tree of the generated problems.

Step 1b. Solve $\tilde{R}_c$, the relaxed version of the problem $\tilde{P}_c$ at hand, using the GrossSimplex and get $\bar{\mathbf{x}}$ and $\tilde{f}_c$ ($= \tilde{\mathbf{c}}^T \bar{\mathbf{x}}$):

$$[\bar{\mathbf{x}}, \tilde{f}_c, \texttt{emptyPolyhedron}] \leftarrow \texttt{GrossSimplex}(\tilde{R}_c)$$

Step 2a. If the LP solver has found that the polyhedron is empty, then prune the sub-tree of $(\tilde{P}_c)$ (according to Pruning Rule 1) by going to Step 1a (without branching $(\tilde{P}_c)$). Otherwise, we have found a new lower value for $\tilde{P}_c$:

$$\tilde{v}_I(\tilde{P}_c) = \tilde{f}_c$$

Step 2b. If $\tilde{v}_I(\tilde{P}_c) \geqslant \tilde{v}_S(\tilde{P})$, then prune the sub-tree under $\tilde{P}_c$ (according to Pruning Rule 2), by going to Step 1a (without branching $\tilde{P}_c$).

Step 2c. If $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$ and all components of $\bar{\mathbf{x}}$ that must be integer are actually ϵ-integer (i.e., $\bar{\mathbf{x}}$ is feasible), then we have found a better upper bound estimate. Thus we can update the value of $\tilde{v}_S(\tilde{P})$ as:

$$\tilde{v}_S(\tilde{P}) = \tilde{v}_I(\tilde{P}_c).$$

In addition we set $\mathbf{x}_{opt} = \bar{\mathbf{x}}$ and $\tilde{f}_{opt} = \tilde{v}_I(\tilde{P}_c)$. Then we also prune the sub-tree under $(\tilde{P}_c)$ (according to Pruning Rule 3) by going to Step 1a (without branching $(\tilde{P}_c)$).

Step 3. If $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$ but *not* all components of $\bar{\mathbf{x}}$ that must be integer are actually ϵ-integer, we have to branch. Select the component $\bar{x}_t$ of $\bar{\mathbf{x}}$ having the greatest fractional part, among all the components that must be integer. Create two new nodes (i.e., problems) with a new constraint for this variable, one with a new $\leqslant$ constraint for the rounded down value of $\bar{x}_t$ and another with a new $\geqslant$ constraint for the rounded up value of $\bar{x}_t$. Let us call the two new problems $\tilde{P}_l$ and $\tilde{P}_r$ and put them at the tail of the queue of the problems to be solved, then goto Step 1a.

Step 4. End of the algorithm.

5.7 Experimental results

In this section we introduce three LMOMILP test problems having known solution, then we verify that the GrossBB combined with the GrossSimplex is able to solve them. Additional examples can be found in [2].

5.7.1 Test problem 1: the "kite" in 2D

This problem is a little variant to problem T_0 described above. The main difference is the addition of the integrality constraint. In addition, the second constraint $2x_1 + 3x_2 \leqslant 210$ has been changed into $2x_1 + 3x_2 \leqslant 210 + 2.5$). Finally observe how this formulation does not involve slack variables:

$$
\begin{array}{lll}
\text{LexMax} & 8x_1 + 12x_2,\ 14x_1 + 10x_2,\ x_1 + x_2 & \\
\text{s.t.} & 2x_1 + 1x_2 \leqslant 120 & \\
& 2x_1 + 3x_2 \leqslant 210 + 2.5 & \\
& 4x_1 + 3x_2 \leqslant 270 & \\
& x_1 + 2x_2 \geqslant 60 & \\
& -200 \leqslant x_1, x_2 \leqslant +200, \qquad \mathbf{x} \in \mathbb{Z}^n &
\end{array}
\qquad T_1
$$

The polygon $\mathcal{S}$ associated to problem T_1 is shown in Fig. 5.2 (left sub-figure). The integer points (feasible solutions) are shown as black spots, while in light grey we have provided the domain of the relaxed problem (without the integer constraints). It can be seen that the first objective vector $\mathbf{c}^1 = [8,\ 12]^T$ is orthogonal to segment $[\alpha,\ \beta]$ ($\alpha = (0,\ 70.83), \beta = (28.75,\ 51.67)$) shown in the same figure. All the nearest integer points parallel to this segment are optimal for the first objective (see the right sub-figure in Fig. 5.2). Since the solution is not unique, there is the chance to try to improve the second objective vector ($\mathbf{c}^2 = [14, 10]^T$).

Let see what happens when we solve this problem using the GrossSimplex-based GrossBB algorithm.

Since the T_1 problem is $LexMax$-formulated, we have to feed $-\tilde{\mathbf{c}}$ to the GrossSimplex-based GrossBB algorithm.

Initialize $\tilde{v_S}(\tilde{P}) = ①$, $\mathbf{x}_{opt} = [\ \ ]$, $\tilde{f}_{opt} = ①$ and insert T_1 into a queue of the sub-problems that must be solved.

Iteration 1 The GrossBB extracts from the queue of problems to be solved the only one present, and denotes it as the current problem: $\tilde{P}_c \equiv T_1$). Then the algorithm solves its relaxed version: the solution of $\tilde{R}_c$ is $\bar{\mathbf{x}} = [28.7500,\ 51.6667]^T$, with $\tilde{v_I}(\tilde{P}_c) = -850①^0 - 919.167①^{-1} - 80.4167①^{-2}$. In this case we have to branch using the component having the highest fractional part (among the variables with integer restrictions, of course). In this case, it is the first component, and thus the new sub-problem on the left $\tilde{P}_l$ will have the additional constraint $x_1 \leqslant 28$, while the new on the right $\tilde{P}_r$ will have the additional constraint $x_1 \geqslant 29$. This split makes the current solution $[28.7500, 51.6667]^T$ not optimal neither for problems $\tilde{P}_l$ nor for $\tilde{P}_r$ (see Fig. 5.3).

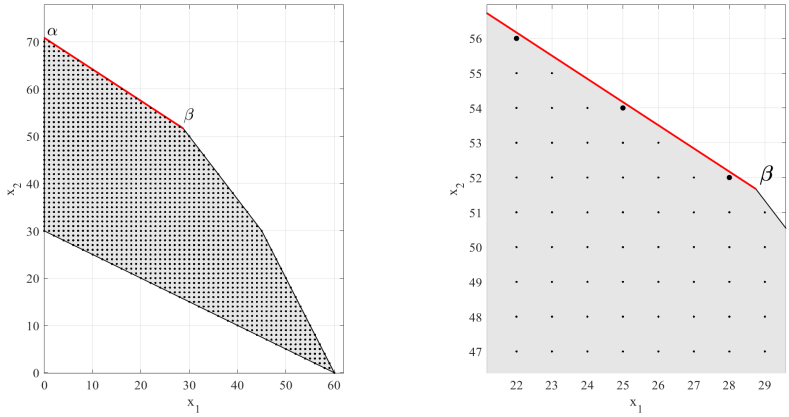

Fig. 5.2 An example in two dimensions with three objectives. The black points on the left figure are all the feasible solutions. All the nearest integer points parallel to the segment $[\alpha, \beta]$ (there are many), are optimal for the first objective, while point $(28, 52)$ is the unique lexicographic optimum for the given problem (i.e., considering the second objective too). The third objective plays no role in this case. On the right, a zoom around point β is provided, with some optimal solutions for the first objective highlighted (the ones with a bigger black spot).

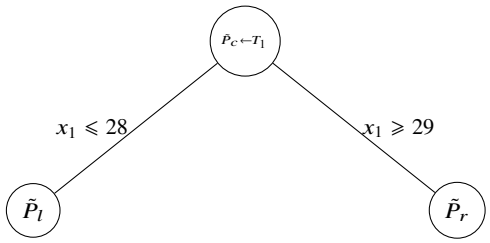

Fig. 5.3 Situation at the end of iteration 1, for problem T_1.

Iteration 2 At this step the queue is composed by $[\tilde{P}_l, \tilde{P}_r]$, the problems generated in the previous iteration. The GrossBB extracts now the next problem from the top of the queue (*breadth-first* visit), namely $[\tilde{P}_l$ and denotes it as $\tilde{P}_c$, the current problem to solve. The optimal solution for the relaxed problem $\tilde{R}_c$ is $\bar{\mathbf{x}} = [28.25, 52]^T$, with $\tilde{v_l}(\tilde{P}_c) = -850①^0 - 915.5①^{-1} - 80.25①^{-2}$. We have to branch again, as we did on iteration 1, a thus a new left and right problems will be generated and added to the queue. The new left problem will have the additional constraint $x_1 \leqslant 28$, while the new right problem will have the additional constraint $x_1 \geqslant 29$. The length of the queue is now 3.

Iteration 3 Extract the next problem from the top of the queue and denote it as $\tilde{P}_c$. The optimal solution of $\tilde{R}_c$ is $\bar{\mathbf{x}} = [29.25, 51]^T$ and the associated $\tilde{v_I}(\tilde{P}_c) = -846①^0 - 919.5①^{-1} - 80.25①^{-2}$. We have to branch, a new left and right problem will be generated, both will be added to the queue. The left problem will have the additional constraint $x_1 \leqslant 29$, while the new on the right $\tilde{P}_r$ will have the additional constraint $x_1 \geqslant 30$. The length of the queue is now 4.

Iteration 4 Extract the next problem from the queue and indicate it as $\tilde{P}_c$. Solve $\tilde{R}_c$, the relaxation of $\tilde{P}_c$, using the GrossSimplex. Since $\tilde{R}_c$ has an empty feasible region, prune this node by applying the first pruning rule. The length of the queue is now 3.

Iteration 5 Extract the next problem from the top of the queue and denote it as $\tilde{P}_c$. The optimal solution of $\tilde{R}_c$ is $\bar{\mathbf{x}} = [28, 52.1667]^T$ and the associated $\tilde{v_I}(\tilde{P}_c) = -850①^0 - 913.6667①^{-1} - 80.1667①^{-2}$. We have to branch: a new left and right problems will be generated and added to the queue. The left problem will have the additional constraint $x_1 \leqslant 52$, while the right one will have the additional constraint $x_1 \geqslant 53$. The length of the queue is now 4.

Iteration 6 Extract the next problem from the queue and indicate it as $\tilde{P}_c$. This time the GrossSimplex returns an integer solution, i.e., a feasible solution for the LMOMILP initial problem T_1:

$$\bar{\mathbf{x}} = [30, 50]^T \quad \text{and} \quad \tilde{v_I}(\tilde{P}_c) = -840①^0 - 920①^{-1} - 80①^{-2}$$

Since $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$, then we can update $\tilde{v}_S(\tilde{P}) = \tilde{v}_I(\tilde{P}_c)$, $\mathbf{x}_{opt} = \bar{\mathbf{x}}$. Finally we can prune this node, according to the third pruning rule.

Iteration 7 Extract the next problem from the queue and indicate it as $\tilde{P}_c$. Again the GrossSimplex returns an integer solution:

$$\bar{\mathbf{x}} = [29, 51]^T \quad \text{and} \quad \tilde{v_I}(\tilde{P}_c) = -844①^0 - 916①^{-1} - 80①^{-2}$$

Since $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$, then we can update $\tilde{v}_S(\tilde{P}) = \tilde{v}_I(\tilde{P}_c)$, $\mathbf{x}_{opt} = \bar{\mathbf{x}}$. Finally we can prune this node, according to the third pruning rule.

Iteration 8 Extract the next problem from the top of the queue and indicate it as $\tilde{P}_c$. The optimal solution of $\tilde{R}_c$ is $\bar{\mathbf{x}} = [26.75, 53]^T$, with $\tilde{v_I}(\tilde{P}_c) = -850①^0 - 904.5①^{-1} - 79.75①^{-2}$. We have to branch: a new left and right problems will be generated and added to the queue. The left problem will have the additional constraint $x_1 \leqslant 26$, while the new on the right $\tilde{P}_r$ will have the additional constraint $x_1 \geqslant 27$. The length of the queue is now 4.

Iteration 9 Extract the next problem from the queue and designate it as the current problem $\tilde{P}_c$. Solve its relaxation, using the GrossSimplex. In this case the returned solution is feasible for the initial LMOMILP problem T_1, because it has all integral components:

$$\bar{\mathbf{x}} = [28,\ 52]^T \quad \text{and} \quad \tilde{v_I}(\tilde{P}_c) = -848①^0 - 912①^{-1} - 80①^{-2}.$$

Since $\tilde{v}_I(\tilde{P}_c) < \tilde{v}_S(\tilde{P})$, then update both $\tilde{v}_S(\tilde{P}) = \tilde{v}_I(\tilde{P}_c)$ and $\mathbf{x}_{opt} = \bar{\mathbf{x}}$. Finally prune this node by applying the third pruning rule.

Iteration 10 Extract the next problem from the queue and indicate it as $\tilde{P}_c$. Solve $\tilde{R}_c$, the relaxation of $\tilde{P}_c$, using the GrossSimplex. Since $\tilde{R}_c$ has an empty feasible region, prune this node by applying the first pruning rule.

Iterations 11-79 The GrossBB algorithm is not able to find a better solution than the $\bar{\mathbf{x}} = [28,\ 52]^T$ already found, but continues to branch and explore the tree, until only two nodes remain in the queue. The processing of the last two nodes is discussed in the last two iterations 80 and 81, below.

Iteration 80 Extract the next problem from the queue and indicate it as $\tilde{P}_c$. Solve $\tilde{R}_c$ using the GrossSimplex. Since $\tilde{R}_c$ has an empty feasible region, prune this node by applying the first pruning rule.

Iteration 81 At this point there is one last unsolved problem from the queue. Extract this problem and indicate it as $\tilde{P}_c$. The optimal solution of $\tilde{R}_c$ is:

$$\bar{\mathbf{x}} = [1,\ 70]^T, \quad \text{with} \quad \tilde{v_I}(\tilde{P}_c) = -848①^0 - 714①^{-1} - 71①^{-2}.$$

Since $\tilde{v_I}(\tilde{P}_c) \geq \tilde{v}_S(\tilde{P})$, prune this last node according to the third pruning rule. Being now the tree empty, the GrossBB algorithm stops according to the first terminating condition and returns the optimal solution found so far: $\mathbf{x}_{opt} = [28,\ 52]^T$. The optimal value of the objective function is $\tilde{\mathbf{c}}^{\mathbf{T}}\mathbf{x}_{opt} = 848①^0 + 912①^{-1} + 80①^{-2}$.

Table 5.2 provides a synthesis with the most interesting iterations performed by the GrossBB.

Table 5.2 Iterations performed by GrossSimplex-based GrossBB algorithm on problem T_1

Iteration	result at node(iteration)
Initialize	- $\tilde{v_S}(\tilde{P}) = ①$
	- Queue len. 1 (add the root problem to the queue)
1	$\tilde{v_I}(\tilde{P}_c)$: $-850①^0 - 919.167①^{-1} - 80.4167①^{-2}$. Queue length : 0
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 2
	- $\tilde{\Delta} = 100①^0 + 100①^{-1} + 100①^{-2}$
2	$\tilde{v_I}(\tilde{P}_c)$: $-850①^0 - 915.5①^{-1} - 80.25①^{-2}$. Queue length: 1
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 3
	- $\tilde{\Delta} = 100①^0 + 100①^{-1} + 100①^{-2}$
3	$\tilde{v_I}(\tilde{P}_c)$: $-846①^0 - 919.5①^{-1} - 80.25①^{-2}$. Queue length: 2
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 4
	- $\tilde{\Delta} = 100①^0 + 100①^{-1} + 100①^{-2}$
4	prune node: rule 1, empty feasible region. Queue length: 3
5	$\tilde{v_I}(\tilde{P}_c)$: $-850①^0 - 913.667①^{-1} - 80.1667①^{-2}$. Queue length: 2
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 4
	- $\tilde{\Delta} = 100①^0 + 100①^{-1} + 100①^{-2}$
6	$\tilde{v_I}(\tilde{P}_c)$: $-840①^0 - 920①^{-1} - 80①^{-2}$. Queue length: 3
	- A feasible solution has been found: $\mathbf{x}_{opt} = [30,\ 50]^T$
	- update $\tilde{v_S}(\tilde{P}) = \tilde{v_I}(\tilde{P}_c)$, prune node: rule 3
	- $\tilde{\Delta} = 0.0119048①^0 - 0.00688406①^{-1} + 0.00208333①^{-2}$
7	$\tilde{v_I}(\tilde{P}_c)$: $-844①^0 - 916①^{-1} - 80①^{-2}$. Queue length: 2
	- A feasible solution has been found: $\mathbf{x}_{opt} = [29,\ 51]^T$
	- update $\tilde{v_S}(\tilde{P}) = \tilde{v_I}(\tilde{P}_c)$, prune node: rule 3
	- $\tilde{\Delta} = 0.354191①^0 - 0.127528①^{-1} + 0.104058①^{-2}$
8	$\tilde{v_I}(\tilde{P}_c)$: $-850①^0 - 904.5①^{-1} - 79.75①^{-2}$. Queue length: 1
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 3
	- $\tilde{\Delta} = 0.007109①^0 - 0.00254731①^{-1} + 0.00208333①^{-2}$
9	$\tilde{v_I}(\tilde{P}_c)$: $-848①^0 - 912①^{-1} - 80①^{-2}$. Queue length: 2
	- A feasible solution has been found: $\mathbf{x}_{opt} = [28,\ 52]^T$
	- update $\tilde{v_S}(\tilde{P}) = \tilde{v_I}(\tilde{P}_c)$, prune node: rule 3
	- $\tilde{\Delta} = 0.00235849①^0 - 0.00822368①^{-1} - 0.003125①^{-2}$
10	prune node: rule 1, empty feasible region. Queue length: 1
...	... ...
80	prune node: rule 1, empty feasible region. Queue length: 1
81	$\tilde{v_I}(\tilde{P}_c)$: $-848①^0 - 714①^{-1} - 71①^{-2}$. Queue length: 0
	- $\tilde{v_I}(\tilde{P}_c) \geqslant \tilde{v_S}(\tilde{P})$ prune node: rule 2
result	Iteration 81. Optimization ended. Optimal solution found:
	$\mathbf{x}_{opt} = [28,\ 52]^T$
	$\tilde{f}_{opt} = -848①^0 - 912①^{-1} - 80①^{-2}$
	$\tilde{\Delta} = 0①^0 + 0①^{-1} + 0①^{-2}$

5.7.2 Test problem 2: the unrotated “house" in 3D

This illustrative example is in three dimensions with three objectives:

$$
\begin{array}{ll}
\text{LexMax} & x_1, -x_2, -x_3 \\
\text{s.t.} & -10.2 \leqslant x_1 \leqslant 10.2 \\
& -10 \leqslant x_2 \leqslant 10.2 \\
& -10.2 \leqslant x_3 \leqslant 10.2 \\
& -x_1 - x_2 \leqslant 2 \\
& -x_1 + x_2 \leqslant 2 \\
& -20 \leqslant x_i \leqslant 20,\ i = 1, \ldots, 3, \qquad \mathbf{x} \in \mathbb{Z}^3
\end{array}
\qquad T_2
$$

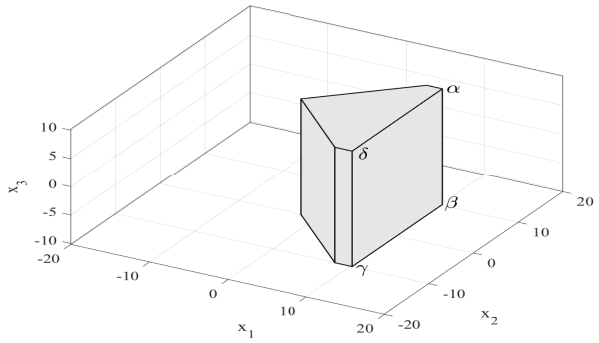

Fig. 5.4 A 3D unrotated “house" problem.

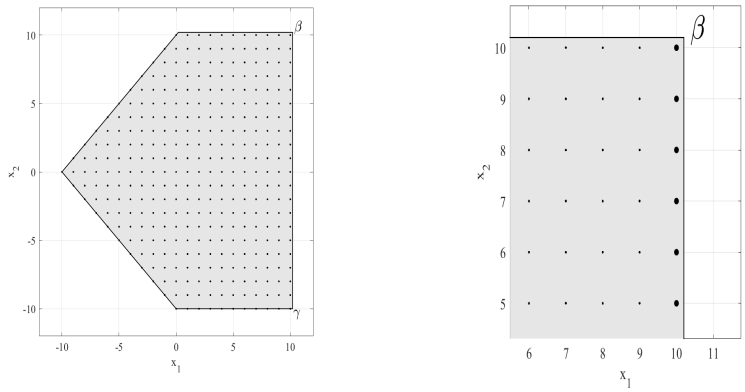

Fig. 5.5 Section view of Fig. 5.4 with $x_3 = -10$ (left) and its top-right zoom (right).

with the domain being the cube shown in Fig. 5.4. It can be immediately seen that by considering the first objective alone (maximize x_1), all the nearest integer points parallel to square having vertices α, β, γ, δ (see Fig. 5.4) are optimal for the first objective function. Since the optimum is not unique, the second objective function can be considered in order to improve it without deteriorating the first objective. Then, all the integer points near to the segment $[\beta, \gamma]$ are all optimal for the second objective too (see Fig. 5.5, which provides the plant-view of Fig. 5.4 with $x_3 = -10$). Again, the optimum in not unique, and thus we can consider the third objective, which allows us to select the nearest integer point to γ as the unique solution that maximizes all the three objectives. In particular, point $[10, -10, -10]$ is the lexicographic optimum to the given problem.

The problem can be solved with GrossBB algorithm, as shown in Tab. 5.3. The solution $\mathbf{x}_{opt} = [10, -10, -10]^T$ is actually found after 5 iterations. The optimal value of the objective function can be computed as $\tilde{\mathbf{c}}^T \mathbf{x}_{opt} = 10①^0 + 10①^{-1} + 10①^{-2}$.

Table 5.3 Iterations performed by GrossBB algorithm on test problem T_2

Iteration	result at node(iteration)
Initialize	- $\tilde{v_S}(\tilde{P}) = ①$
	- Queue len. 1 (add the root problem to the queue)
1	$\tilde{v_I}(\tilde{P}_c)$: $-10.2①^0 - 10①^{-1} - 10.2①^{-2}$. Queue length: 0
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 2
	- $\tilde{\Delta}$ $100①^0 + 100①^{-1} + 100①^{-2}$
2	prune node: rule 1, empty feasible region. Queue length: 1
3	$\tilde{v_I}(\tilde{P}_c)$: $-10①^0 - 10①^{-1} - 10.2①^{-2}$. Queue length: 0
	- no pruning rules applied, branch $\tilde{P}_c$ in two sub-problems. Queue length: 2
	- $\tilde{\Delta}$ $100①^0 + 100①^{-1} + 100①^{-2}$
4	$\tilde{v_I}(\tilde{P}_c)$: $-10①^0 - 10①^{-1} - 10①^{-2}$. Queue length: 1
	- A feasible solution has been found: $x_{opt} = [10, -10, -10]^T$
	- update $\tilde{v_S}(\tilde{P}) = \tilde{v_I}(\tilde{P}_c)$, prune node: rule 3
	- $\tilde{\Delta}$: $0①^0 + 0①^{-1} + 0.02①^{-2}$
5	prune node: rule 1, empty feasible region. Queue length: 0
result	Iteration 5. Optimization ended. Optimal solution found:
	$\mathbf{x}_{opt} = [10, -10, -10]^T$
	$\tilde{f}_{opt} = -10①^0 - 10①^{-1} - 10①^{-2}$
	$\tilde{\Delta} = 0①^0 + 0①^{-1} + 0①^{-2}$

5.7.3 Test problem 3: the rotated “house" in 5D

Algorithm 3 shows how to add a small rotation along the axis perpendicular to the plane containing the first two variables x_1 and x_2, for the “house" problem seen in previous example, after generalizing it to the n-dimensional case.

The problem consists of the lexicographic optimization of x_1, $-x_2$, ..., $-x_5$. The method used to generate a randomly rotated benchmark is shown on Algorithm 3 (the generated rotation matrix $\mathbf{Q}$ is reported in Appendix A of [2]). After the rotation, a lower bound and an upper bound were added:

$$-2\rho \leqslant x_i \leqslant 2\rho,\ i = 1, ..., 5.$$

Thus the following problem ($\mathbf{A}'$,$\mathbf{b}'$) has been generated:

$$\begin{aligned} \text{LexMax} \quad & x_1, -x_2, ..., -x_5 \\ \text{s.t.} \quad & \left\{\mathbf{x}' \in \mathbb{Z}^5 : \mathbf{A}'\mathbf{x}' \leqslant \mathbf{b}'\right\} \end{aligned} \qquad T_3$$

where $\mathbf{C}'$, $\mathbf{A}'$ and vector $\mathbf{b}'$ are reported in Appendix A of [2].

The lexicographic optimum for this problem is:

$$\mathbf{x}_{opt} = [1000,\ -999,\ -1000,\ -1000,\ -1000]^T.$$

The new problem can be solved with GrossBB algorithm. After 11 steps, the GrossBB finds the correct lexicographic optimum (more details in [2], Tab. 3, Appendix B).

5.8 Conclusions

To conclude this chapter, let us recall that we have shown the application of Grossone to solve lexicographic multi-objective linear programming problems and their mixed-integer counterparts. We have proved that the use of Grossone allows to give an infinitely lower weights to lower-priority objectives, without the need to specify the value of the finite weight M to use. As a future work, we are planning to implement a Grossone-based Branch-and-Cut algorithm, to speedup the convergence of the GrossBB algorithm presented here.

Algorithm 3 Generation of a randomly rotated "house" problem in $\mathbb{R}^n$ dimensions

Step 1. Let $\{\mathbf{Ax} \leqslant \mathbf{b},\ \mathbf{x} \in \mathbb{Z}^n\}$ be the initial, unrotated problem, in n-dimensions. The problems is formulated as follow (ρ is a parameter that controls the size of the house):

$$\begin{array}{ll} \text{LexMax} & x_1,\ -x_2,\ \ldots,\ -x_n \\ \text{s.t.} & -\rho + 0.2 \leqslant x_1 \leqslant \rho + 0.2, \\ & -\rho \qquad\quad \leqslant x_2 \leqslant \rho + 0.2 \\ & -\rho + 0.2 \leqslant x_i \leqslant \rho + 0.2,\ i = 3, \ldots, n \\ & -x_1 - x_2 \leqslant 2 \\ & -x_1 + x_2 \leqslant 2 \\ & \mathbf{x} \in \mathbb{Z}^n \end{array}$$

Step 2. Use as rotation matrix $\mathbf{Q}$ with a random little rotation:

```
rA = 0.0002;

rB = 0.0005;

φ = (rB-rA).*rand(1) + rA;
```

$$\mathbf{Q} = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 & 0 & .. & 0 \\ -\sin(\phi) & \cos(\phi) & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Step 3. Rotate the polytope: $\mathbf{A'} = \mathbf{AQ}$ ($\mathbf{b}$ and $\mathbf{C}$ does not change under rotations: $\mathbf{b'} = \mathbf{b}$ and $\mathbf{C'} = \mathbf{C}$) and then add these additional constraints as lower and upper bound for every variables to $\mathbf{A'}$ (they are twice the size of the house, in order to fully contain it):

$$-2\rho \leqslant x_i \leqslant 2\rho,\ i = 1, \ldots, n$$

Step 4 For the unrotated problem the optimal integer solution is $\mathbf{x}_{opt} = [\rho, -\rho, -\rho, -\rho, \ldots, -\rho]^T$.
When a rotation is applied, if the rotation is between a sufficiently small range of angles the optimal solution is: $\mathbf{x}_{opt} = [\rho, 1-\rho, -\rho, -\rho, \ldots, -\rho]^T$.
The optimal value is computed as: $\tilde{f}_{opt} = \tilde{\mathbf{c}}^T \mathbf{x}_{opt}$, where $\tilde{\mathbf{c}}$ is derived from $\mathbf{C}$.

Appendix

In this appendix we provide the proofs of Lemmas 1-3, taken from [6], and reported also here for the sake of self-completeness of this exposition.

Proof (lemma1) Since the objective function of problem P4 is linear in $\mathbf{x}$, the associated level sets ($\tilde{\mathbf{c}}^T\mathbf{x} = \tilde{v}$) are hyper-planes. Thus the maximum, for a bounded and non-empty polyhedron $\mathcal{S}$ describing the domain of the problem must be on a vertex (or belong to the convex hull of the optimal vertices), for the same reasons for which the maximum of standard single-objective LP problems is located on a vertex or in the convex hull of the optimal vertices. In this case, similarly to standard LP, it follows that: (i) not all the vertices of P4 are optimal, and (ii) not all the basic solutions are vertices.

The next lemma states that all the optimal solutions of P1 reach the same (Gross-scalar) objective value for problem P4.

Proof (Lemma 2) Let $\mathbf{x}^*$ be a generic optimal solution belonging to Ω(P1), then from (5.4) we have that the objective value of the problem P4 associated to it is:

$$\tilde{\mathbf{c}}^T\mathbf{x}^* = (\mathbf{c}^{1T}\mathbf{x}^*)①^0 + (\mathbf{c}^{2T}\mathbf{x}^*)①^{-1} + ... + (\mathbf{c}^{rT}\mathbf{x}^*)①^{-r+1}.$$

We can observe that:

$$\mathbf{c}^{iT}\mathbf{x}^* = k^i \in \mathbb{R}, \quad i \in \{1, ..., r\},$$

due to the fact that $\mathbf{x}^*$ is optimal for P1 and this problem has been formulated using purely finite numbers only. Thus we have

$$\tilde{\mathbf{c}}^T\mathbf{x}^* = k^1①^0 + k^2①^{-1} + ... + k^r①^{-r+1} = \tilde{v}.$$

The next lemma is the last step in preparing the proof of the equivalence between P1 and P4.

Proof (Lemma 3) Let $\hat{\mathbf{x}}$ be a vertex of S which is not optimal for P1. Thus, there exists an index $q \in \{1, ..., r\}$ such that

$$\mathbf{c}^{iT}\hat{\mathbf{x}} = \mathbf{c}^{iT}\mathbf{x}^*, \quad \forall i \in \{1, ..., q-1\},$$

but

$$\mathbf{c}^{qT}\hat{\mathbf{x}} \ < \ \mathbf{c}^{qT}\mathbf{x}^*.$$

This implies that

$$\tilde{\mathbf{c}}^T\mathbf{x}^* - \tilde{\mathbf{c}}^T\hat{\mathbf{x}} = \sum_{i=q}^{r} ①^{-i+1}(\mathbf{c}^{iT}\mathbf{x}^* - \mathbf{c}^{iT}\hat{\mathbf{x}}).$$

The expression above can be also expanded as follows

$$\tilde{\mathbf{c}}^T\mathbf{x}^* - \tilde{\mathbf{c}}^T\hat{\mathbf{x}} = ①^{-q+1}(\mathbf{c}^{qT}\mathbf{x}^* - \mathbf{c}^{qT}\hat{\mathbf{x}}) + ①^{-(q+1)+1}(\mathbf{c}^{(q+1)T}\mathbf{x}^* - \mathbf{c}^{(q+1)T}\hat{\mathbf{x}}) + \ldots$$
$$+ ①^{-r+1}(\mathbf{c}^{rT}\mathbf{x}^* - \mathbf{c}^{rT}\hat{\mathbf{x}}).$$

Since $①^{-q+1}(\mathbf{c}^{qT}\mathbf{x}^* - \mathbf{c}^{qT}\hat{\mathbf{x}}) > 0$, it follows that $\left(\tilde{\mathbf{c}}^T\mathbf{x}^* - \tilde{\mathbf{c}}^T\hat{\mathbf{x}}\right)$ is strictly positive too, since r is finite. Indeed, adding a finite number of infinitesimal contributions of orders of ① higher than $-q+1$ will keep the sum strictly positive, even when these contributions are negative in sign, due to the property of Grossone:

$$\left|①^{-q+1}(\mathbf{c}^{qT}\mathbf{x}^* - \mathbf{c}^{qT}\hat{\mathbf{x}})\right| > \left|①^{-q}(\mathbf{c}^{(q+1)T}\mathbf{x}^* - \mathbf{c}^{(q+1)T}\hat{\mathbf{x}})\right| + \ldots + \left|①^{-r+1}(\mathbf{c}^{rT}\mathbf{x}^* - \mathbf{c}^{rT}\hat{\mathbf{x}})\right|.$$

References

[1] Amodio, P., Iavernaro, F., Mazzia, F., Mukhametzhanov, M.S., Sergeyev, Y.D.: A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic. Mathematics and Computers in Simulation **141**, 24–39 (2017)

[2] Cococcioni, M., Cudazzo, A., Pappalardo, M., Sergeyev, Y.D.: Solving the lexicographic multi-objective mixed-integer linear programming problem using branch-and-bound and grossone methodology. Communications in Nonlinear Science and Numerical Simulation **84**, 105177 (2020)

[3] Cococcioni, M., Fiaschi, L.: The Big-M method with the numerical infinite M. Optimization Letters **15**, 2455–2468 (2021)

[4] Cococcioni, M., Fiaschi, L., Lambertini, L.: Non-archimedean zero-sum games. Journal of Computational and Applied Mathematics **393**, 113483 (2021)

[5] Cococcioni, M., Pappalardo, M., Sergeyev, Y.D.: Towards lexicographic multi-objective linear programming using grossone methodology. In: Y.D. Sergeyev, D.E. Kvasov, F. Dell'Accio, M.S. Mukhametzhanov (eds.) Proc. of the 2nd Intern. Conf. "Numerical Computations: Theory and Algorithms", vol. 1776, p. 090040. AIP Publishing, New York (2016)

[6] Cococcioni, M., Pappalardo, M., Sergeyev, Y.D.: Lexicographic multi-objective linear programming using grossone methodology: Theory and algorithm. Applied Mathematics and Computation **318**, 298–311 (2018)

[7] De Cosmis, S., De Leone, R.: The use of grossone in mathematical programming and operations research. Applied Mathematics and Computation **218**(16), 8029–8038 (2012)

[8] De Leone, R.: Nonlinear programming and grossone: Quadratic programming and the role of constraint qualifications. Applied Mathematics and Computation **318**, 290–297 (2018)

[9] De Leone, R., Fasano, G., Roma, M., Sergeyev, Y.D.: Iterative grossone-based computation of negative curvature directions in large-scale optimization. Jour-

nal of Optimization Theory and Applications **186**(2), 554–589 (2020)
[10] De Leone, R., Fasano, G., Sergeyev, Y.D.: Planar methods and grossone for the conjugate gradient breakdown in nonlinear programming. Computational Optimization and Applications **71**, 73–93 (2018)
[11] Fiaschi, L., Cococcioni, M.: Numerical asymptotic results in game theory using Sergeyev's Infinity Computing. Int. Journal of Unconventional Computing **14**(1), 1–25 (2018)
[12] Fiaschi, L., Cococcioni, M.: Non-archimedean game theory: A numerical approach. Applied Mathematics and Computation **409**, 125356 (2021)
[13] Gaudioso, M., Giallombardo, G., Mukhametzhanov, M.S.: Numerical infinitesimals in a variable metric method for convex nonsmooth optimization. Applied Mathematics and Computation **318**, 312–320 (2018)
[14] Isermann, H.: Linear lexicographic optimization. OR Spektrum **4**, 223–228 (1982)
[15] Lai, L., Fiaschi, L., Cococcioni, M.: Solving mixed pareto-lexicographic many-objective optimization problems: The case of priority chains. Swarm and Evolutionary Computation **55**, 100687 (2020)
[16] Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Handling priority levels in mixed pareto-lexicographic many-objective optimization problems. In: H. Ishibuchi, Q. Zhang, R. Cheng, K. Li, H. Li, H. Wang, A. Zhou (eds.) Evolutionary Multi-Criterion Optimization, pp. 362–374. Springer International Publishing, Cham (2021)
[17] Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Solving mixed pareto-lexicographic many-objective optimization problems: The case of priority levels. IEEE Transactions on Evolutionary Computation (2021). `http://dx.doi.org/10.1109/TEVC.2021.3068816`
[18] Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Pure and Mixed Lexicographic-Paretian Many-Objective Evolutionary Optimization: State of the Art. Natural Computing (2022). *Submitted.*
[19] Pappalardo, M., Passacantando, M.: Ricerca Operativa. Pisa University Press (2012)
[20] Pourkarimi, L., Zarepisheh, M.: A dual-based algorithm for solving lexicographic multiple objective programs. European Journal of Operational Research **176**, 1348–1356 (2007)
[21] Rizza, D.: Supertasks and numeral systems. In: Proc. of the 2nd Intern. Conf. "Numerical Computations: Theory and Algorithms", vol. 1776. AIP Publishing, New York (2016). URL `https://doi.org/10.1063/1.4965369`. 090005
[22] Rizza, D.: A study of mathematical determination through Bertrand's Paradox. Philosophia Mathematica **26**(3), 375–395 (2018)
[23] Rizza, D.: Numerical methods for infinite decision-making processes. Int. Journal of Unconventional Computing **14**(2), 139–158 (2019)
[24] Sergeyev, Y.D.: Solving ordinary differential equations by working with infinitesimals numerically on the Infinity Computer. Applied Mathematics and Computation **219**(22), 10668–10681 (2013)

[25] Sergeyev, Y.D.: The olympic medals ranks, lexicographic ordering, and numerical infinities. The Mathematical Intelligencer **37**(2), 4–8 (2015)
[26] Sergeyev, Y.D.: Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems. EMS Surv. Math. Sci. **4**, 219–320 (2017). DOI 10.4171/EMSS/4-2-3
[27] Sergeyev, Y.D.: Independence of the grossone-based infinity methodology from non-standard analysis and comments upon logical fallacies in some texts asserting the opposite. Foundations of Science **24**(1), 153–170 (2019)
[28] Sergeyev, Y.D., Kvasov, D.E., Mukhametzhanov, M.S.: On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales. Communications in Nonlinear Science and Numerical Simulation **59**, 319–330 (2018)
[29] Sergeyev, Y.D., Mukhametzhanov, M.S., Mazzia, F., Iavernaro, F., Amodio, P.: Numerical methods for solving initial value problems on the Infinity Computer. Int. Journal of Unconventional Computing **12(1)**, 3–23 (2016)
[30] Sherali, H., Soyster, A.: Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples. Journal of Optimization Theory and Applications **39(2)**, 173–186 (1983)
[31] Stanimirovic, I.: Compendious lexicographic method for multi-objective optimization. Facta universitatis - series: Mathematics and Informatics **27**(1), 55–66 (2012)