Computational Management Science (2023) 20:38
https://doi.org/10.1007/s10287-023-00471-1

ORIGINAL PAPER

Solving linear multiplicative programs via branch-and-bound: a computational experience

R. Cambini[1] · R. Riccardi[2] · D. Scopelliti[2]

Received: 6 November 2022 / Accepted: 10 August 2023

Abstract
In this paper, linear multiplicative programs are approached with a branch-and-bound scheme and a detailed computational study is provided. Several underestimation functions are analyzed and various partitioning criteria are presented. A particular class of linear multiplicative programs, useful to solve some applicative bilevel problems, is considered from a theoretical point of view to emphasize an efficient solution method. Detailed results of the computational study are provided to point out the performances provided by using various underestimation functions and partitioning criteria, thus improving some of the results of the current literature.

Keywords Linear multiplicative programs · Branch-and-bound · Global optimization · Nonconvex optimization · Bilevel problems

1 Introduction

In the literature, linear multiplicative programs have been widely studied due to their importance from both theoretical and applicative point of views. These problems, strictly related to quadratic programming and bilinear programming, are used in plant layout design, portfolio and financial optimization, VLSI chip design, robust optimization, network flows (see, e.g., Cambini and Martein 2009; Cambini and Sodini 2008; Cambini and Salvi 2010; Gupta 1995; Horst and Pardalos 1995; Horst

✉ R. Cambini
riccardo.cambini@unipi.it

R. Riccardi
rossana.riccardi@unibs.it

D. Scopelliti
domenico.scopelliti@unibs.it

[1] Dipartimento di Economia e Management, Università di Pisa, Via C. Ridolfi 10, 56124 Pisa, Italy

[2] Dipartimento di Economia e Management, Università degli Studi di Brescia, Via S. Faustino 74/b, 25122 Brescia, Italy

Published online: 29 August 2023

and Tuy 1996; Horst et al. 2001; Konno and Kuno 1992; McCarl et al. 1977; Mjelde 1983; Ryoo and Sahinidis 2003; Tuy 2016 and references therein).

From the computational point of view, various approaches have been proposed. For instance, Wang et al. (2012) proposed a branch-and-bound algorithm for global minimization of a generalized linear multiplicative programming; Jiao et al. (2012) proposed a branch and bound algorithm based on the computation of subsequent solutions of the series of linear relaxation programming problems; in a very recent paper, Jiao et al. (2023) proposed a branch-reduction-bound algorithm, based on outer space search and branch-and-bound framework.

A second field of literature develops methods based on an eigenvectors approach. The eigenvectors approach has been widely used despite actually it has some drawbacks. It is well known that the eigen-decomposition of a quadratic function is an heavy task from both a computational and a numerical point of view and it does not care about the particular structure of linear multiplicative functions. For all these reasons, in the recent years, various papers have introduced new procedures developed without the use of eigen-decompositions (see, e.g., Shen et al. 2020, 2022; Wang et al. 2012; Zhou et al. 2015).

In this paper, various underestimation functions to be used in the branch and bound procedure are introduced and discussed both with the eigenvectors approach and without it. Then, a full computational test is provided to highlight the best performing functions.

In addition, a motivating example is presented in the last part of the paper. Special structures of multiplicative problems, in facts, arise in several bilevel programming problems of *leader-follower type* (see Dempe 2020, for example). Some underestimation functions introduced in the paper can be used to improve the algorithmic procedure adopted to tackle this class of problems.

The main contributions of this paper are:

- To describe a *unified framework* for dealing with linear multiplicative programs from both a theoretical and computational point of view;
- To propose various *underestimation functions* to be used in the branch and bound procedure: quadratic, linear, difference of two convex functions (D.C. functions), and eigenvectors based underestimation functions;
- To perform a *detailed computational test* that compares the different underestimation functions under various partition methods to identify the most promising ones;
- To *characterize a special case* of multiplicative programs strictly related to bilevel optimization and to define the more appropriate underestimation function to solve them.

The paper is organized as follows. In Sect. 2, the main definitions and preliminary results are given. In addition, the criteria for the splitting process of the branch-and-bound approach are analyzed. On this basis, Sect. 3 is devoted to the study of quadratic, eigenvectors based and linear underestimation functions. Then, in Sect. 4, the detailed results of a wide computational experience are provided and fully discussed, giving a detailed view of the computational aspects of the solution method

and improving some of the results of the current literature. Furthermore, Sect. 5 points out the behavior of the proposed underestimation functions in a particular class of linear multiplicative programs very useful in applicative bilevel programming. Finally, a section with the conclusions is given.

2 Definitions and preliminary results

The aim of this section is to define the problem and provide the main preliminary results which will allow the development of the paper. In this light, firstly the problem is defined and then the concept and properties of underestimation functions are given. On this basis, a detailed description of a branch-and-bound scheme to solve the problem is provided, in order to approach it in a unifying framework with respect to different underestimation functions and different splitting criteria. Finally, the quadratic form associated to a linear multiplicative function is recalled as well as the eigenvector-based decomposition of such a quadratic form.

2.1 Definition of the problem

From now on, $\mathbb{R}$ will denote the set of real numbers while $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ will be the affinely extended real number system.

Definition 1 Let P be the following minimization problem:

$$P \ : \ \min_{x \in S} f(x) \, .$$

where $S \subseteq \mathbb{R}^n$ is a nonempty polyhedron defined as

$$S = \left\{ x \in \mathbb{R}^n \ : \ A_{in} x \leqq b_{in}, \ A_{eq} x = b_{eq}, \ l_b \leqq x \leqq u_b \right\},$$

with $A_{in} \in \mathbb{R}^{m \times n}$, $b_{in} \in \mathbb{R}^m$, $A_{eq} \in \mathbb{R}^{r \times n}$, $b_{eq} \in \mathbb{R}^r$, and $l_b, u_b \in \overline{\mathbb{R}}^n$, while $f : \mathbb{R}^n \to \mathbb{R}$ is a linear multiplicative function defined as

$$f(x) = \sum_{i=1}^{p} (c_i^T x + c_{0_i})(d_i^T x + d_{0_i}) + a^T x + a_0 \, ,$$

with $c_i, d_i \in \mathbb{R}^n$, $c_{0_i}, d_{0_i} \in \mathbb{R}$ for all $i \in 1, \ldots, p$ and $a \in \mathbb{R}^n$, $a_0 \in \mathbb{R}$.

2.2 Underestimation functions

The use of suitable underestimation functions of f is needed to solve Problem P by means of a *branch-and-bound approach.*

Definition 2 Let $S \subseteq \mathbb{R}^n$ be nonempty. Let $f : S \to \mathbb{R}$ and $\Phi : S \to \mathbb{R}$. Then, Φ is an *underestimation function* of f if:

$$f(x) \geq \Phi(x) \qquad \forall x \in S\,;$$

moreover, $Er : S \to \mathbb{R}$ is the corresponding *error function* defined as $Er(x) = f(x) - \Phi(x)$.

The following useful properties hold.

Lemma 1 *Let $S \subseteq \mathbb{R}^n$ be nonempty. Let $\Phi_1 : S \to \mathbb{R}$ and $\Phi_2 : S \to \mathbb{R}$ be underestimation functions of $f : S \to \mathbb{R}$, with $Er_1 : S \to \mathbb{R}$ and $Er_2 : S \to \mathbb{R}$ the corresponding error functions. Then, the following properties hold*:

(i) *For all $\lambda \in [0,1]$, Φ_λ defined as $\Phi_\lambda(x) = \lambda\Phi_1(x) + (1-\lambda)\Phi_2(x)$ is an underestimation function of f, with error function defined as $Er_\lambda(x) = \lambda Er_1(x) + (1-\lambda)Er_2(x)$;*
(ii) *Φ defined as $\Phi(x) = \max\left\{\Phi_1(x), \Phi_2(x)\right\}$ is an underestimation function of f, with error function defined as $Er(x) = \min\left\{Er_1(x), Er_2(x)\right\}$.*

Proof For any $x \in S$, it results:
(i) for any $\lambda \in [0,1]$,

$$\Phi_\lambda(x) = \lambda\Phi_1(x) + (1-\lambda)\Phi_2(x) \leq \lambda f(x) + (1-\lambda)f(x) = f(x)$$

and

$$\begin{aligned} Er_\lambda(x) &= f(x) - (\lambda\Phi_1(x) + (1-\lambda)\Phi_2(x)) \\ &= (\lambda f(x) + (1-\lambda)f(x)) - (\lambda\Phi_1(x) + (1-\lambda)\Phi_2(x)) \\ &= \lambda Er_1(x) + (1-\lambda)Er_2(x)\,; \end{aligned}$$

(ii) according to Definition 2, one has $f(x) \geq \max\left\{\Phi_1(x), \Phi_2(x)\right\}$; hence, it follows that

$$Er(x) = f(x) - \max\left\{\Phi_1(x), \Phi_2(x)\right\} = \min\left\{f(x) - \Phi_1(x), f(x) - \Phi_2(x)\right\}.$$

□

The following further result will be used in the next subsection as an *algorithmic stopping criterium.*

Lemma 2 *Let $S \subseteq \mathbb{R}^n$ be nonempty. Let $\Phi : S \to \mathbb{R}$ be an underestimation function of $f : S \to \mathbb{R}$, with $Er : S \to \mathbb{R}$ its error function. If $\bar{x} \in \arg\min_S \Phi$ and $Er(\bar{x}) = 0$, then $\bar{x} \in \arg\min_S f$.*

Proof For all $x \in S$, it results:

$$f(\bar{x}) = \Phi(\bar{x}) + Er(\bar{x}) = \Phi(\bar{x}) \leq \Phi(x) \leq f(x)\,;$$

hence, the thesis follows. □

Finally, it is worth noticing that some underestimation functions of f will be obtained by first rewriting f in D.C. form. Let us recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be in D.C. form if it is expressed as $f(x) = q_1(x) - q_2(x)$, with $q_1 : \mathbb{R}^n \to \mathbb{R}$ and $q_2 : \mathbb{R}^n \to \mathbb{R}$ convex functions (see, e.g., Cambini and Salvi 2009, 2010 and references therein).

In this light, the following result will be useful to deduce underestimation functions of f in the case f is rewritten in D.C. quadratic form.

Lemma 3 *Let* $[y^L, y^U] \subset \mathbb{R}$ *with* $-\infty < y^L < y^U < +\infty$: *for all* $y \in [y^L, y^U]$, *it results*

$$-y^2 \geq -(y^L + y^U)y + y^L y^U.$$

Then, for all $y \in [y^L, y^U]$, *the error given by the use of* $-(y^L + y^U)y + y^L y^U$ *instead of* $-y^2$ *is*

$$Er(y) = (y - y^L)(y^U - y) = \frac{1}{4}(y^U - y^L)^2 - \left(y - \frac{y^L + y^U}{2}\right)^2,$$

with $\frac{1}{4}(y^U - y^L)^2$ *the maximum error obtained at* $y = \frac{y^L + y^U}{2}$.

Proof For all $y \in [y^L, y^U]$, the error given by the use of $-(y^L + y^U)y + y^L y^U$ instead of $-y^2$ is:

$$Er(y) = (-y^2) - (-(y^L + y^U)y + y^L y^U).$$

Being

$$0 \leq (y - y^L)(y^U - y) = -y^2 + (y^L + y^U)y - y^L y^U \quad \text{and}$$
$$-y^L y^U = \frac{1}{4}(y^U - y^L)^2 - \frac{1}{4}(y^L + y^U)^2$$

it yields $(y - y^L)(y^U - y) = \frac{1}{4}(y^U - y^L)^2 - \left(y - \frac{y^L + y^U}{2}\right)^2$ and the thesis follows. □

2.3 A branch-and-bound scheme

In order to solve Problem P by using a branch-and-bound approach (see, e.g., Bajaj and Faruque Hasan 2020; Cambini and Sodini 2005, 2008; Cambini and Salvi 2009, 2010; Fampa et al. 2017; Gerard et al. 2017; Shen et al. 2020 and references therein), the following operative scheme will be considered:

- the feasible region will be iteratively partitioned in multidimensional rectangles,

- the function f will be "relaxed" over the single partitions by means of a *convex underestimation function* Φ,
- the convex "relaxed" subproblems will be solved
- the feasible solution having the smallest value of f will be maintained.

Specifically speaking, such a branch-and-bound approach will be implemented by means of:

- a *Priority Queue* (*PQ*) used to store the single partitions sorted with respect to the *Lower Bound* (*LB*) found minimizing the convex "relaxed" subproblems over the partitions;
- a feasible point *Sol* and a value *UB* corresponding, respectively, to the incumbent best feasible solution found and its image $UB = f(Sol)$;
- a convex underestimation function Φ needed to solve the subproblems *over the various partitions.*

The priority queue *PQ* is used to speed up the solution method (at the cost of memory usage of course) since the partition with smaller *LB* is always known and since it does not need of any periodic "pruning" process (in the "pruning" process of a branch-and-bound scheme, the stored partitions having a *LB* not smaller than *UB* are cancelled since they cannot improve the incumbent best feasible solution).

The following commands are aimed to describe the way the priority queue *PQ* is managed:

- *PQisempty*();
- $LB := PQsmallest()$;
- *PQadd*(*LB*, *partition*, *opt*);
- $[partition, opt] := PQextract()$.

The command *PQisempty* tells whether the *PQ* is empty or not; the command *PQsmallest* provides the smallest LB of the partitions stored in the *PQ*; the command *PQadd* adds to the *PQ* a partition as well as the corresponding *Optimal Solution* (*opt*) and its optimal value $LB = \Phi(opt)$ obtained minimizing Φ over the partition itself. Clearly, *PQ* is such that smaller is the value of the *LB*, the higher the priority of the partition in the *PQ* will be. The last command *PQextract* removes from the *PQ* the partition having the smallest *LB* and provides as the output all the data stored.

Remark 1 If the smallest *LB* is such that $LB \geq UB$, then the partitions in the *PQ* are not able to improve the incumbent best feasible solution *Sol* and hence *PQ* can be emptied (being *PQ* a priority queue, the "pruning" process becomes just a final stopping condition and it is no more periodic).

The following subprocedure "*CheckPartition*()" minimizes the underestimation function Φ over a given partition Π, improves the value of *UB* and stores the data in the *PQ* when the potential relative improvement is such that

$$Relimp(LB, UB) > relTol\,, \tag{1}$$

with

$$Relimp(LB, UB) = \frac{abs(UB - LB)}{\max\{abs(UB), abs(LB)\}}$$

and $relTol > 0$ be a chosen tolerance. According to Remark 1, if $LB \geq UB$ then the partition is not added to the queue and discarded ("pruning").

Subprocedure **CheckPartition**($inputs$: Π)
  if not($isempty(S \cap \Pi)$) *then*
    $opt := \arg\min_{x \in S \cap \Pi} \Phi(x)$; $LB := \Phi(opt)$;
    if $LB < UB$ *then*
      $val := f(opt)$;
      if $val < UB$ *then* $UB := val, Sol := opt$ *end if*;
      if $Relimp(LB, UB) > relTol$ *then*
        $PQadd(LB, partition, opt)$;
      end if;
    end if;
  end if;
end subproc.

The overall solution branch-and-bound scheme is described by the procedure "*Solve*()".

Procedure **Solve**($inputs$: P; $outputs$: Opt, Val)
  Let $PQ := \emptyset, UB := +\infty, Sol := []$;
  Determine the smallest partition Π_0 containing S;
  CheckPartition(Π_0);
  while not(*PQisempty*()) and *PQsmallest*() < *UBdo*
    $[\Pi, opt] := PQextract()$;
    Choose the branching variable to be used in the splitting process;
    Split partition Π accordingly to the chosen branching variable : $\Pi = \Pi_1 \cup \Pi_2$;
    CheckPartition(Π_1), *CheckPartition*(Π_2);
  end while;
  Let $Opt := Sol, Val := UB$;
end proc.

Firstly, variables *PQ*, *UB* and *Sol* are initialized and the starting smallest partition Π_0 containing *S* is found and checked. Then, the iterative phase starts and continues up to either the *PQ* is emptied or the stored partitions cannot improve anymore *UB* ("pruning"). At each iteration, the partition with the smaller *LB* is extracted from the *PQ*, a branching variable is selected, and the partition split accordingly (multi-way branching has been shown to provide poor results, see for example Gerard et al. 2017). The two new partitions are then checked. Finally, at the end of the iterative process, outputs are set.

Remark 2 The value of *UB* is fundamental to improving the performance of the algorithm since it is used to discard the not useful partitions. For this reason, the feasible points found while looking for Π_0 should be used to improve values *UB* and *Sol*.

Notice that the convergence of the proposed method has been widely discussed in the literature (see, e.g., Bajaj and Faruque Hasan 2020; Cambini and Salvi 2009, 2010; Fampa et al. 2017; Gerard et al. 2017; Shen et al. 2020 and references therein). Specifically speaking, since the partitions will be split with respect to values not "close" to its boundaries (see Sect. 4), then the tolerance parameter $relTol > 0$ guarantees that condition (1) in subprocedure "*CheckPartition*()" will become false after a sufficiently large number of iterations. The correctness of the method follows since just feasible solutions are evaluated to improve the incumbent best solution and since the whole feasible region is analyzed. This is known to be an *NP-Hard problem* and in the worst case many local but not global optimal solutions can be found.

Some further choices are finally needed to complete the description of the solution process:

- Which underestimation function $\Phi(x)$ should be used? tight underestimation functions improve the algorithm performance, moreover the underestimation function determines the set of branching variables;
- Which branching variable should be chosen to split the current partition?
- With respect to which value of the branching variable the current partition should be split?

Actually, another fundamental choice has been already made:

- At each iteration, the partition with the smaller *LB* is selected and analyzed.

This criterium is aimed to look for feasible solutions having small values, thus allowing to improve *UB* as much as possible and to increase as much as possible the number of partitions discarded by means of the stopping "pruning" condition.

2.4 A raw approach

Problem P is a particular quadratic (usually indefinite) program since $f(x)$ can be rewritten as:

$$f(x) = \sum_{i=1}^{p} (c_i^T x)(d_i^T x) + \hat{a}^T x + \hat{a}_0 = x^T \hat{Q} x + \hat{a}^T x + \hat{a}_0 \,, \tag{2}$$

with

$$\begin{aligned} \hat{a} &:= a + \sum_{i=1}^{p} (c_i d_{0i} + d_i c_{0i}) \,, \\ \hat{a}_0 &:= a_0 + \sum_{i=1}^{p} c_{0i} d_{0i} \,, \\ \hat{Q} &:= \frac{1}{2} \sum_{i=1}^{p} \left(c_i d_i^T + d_i c_i^T \right) . \end{aligned}$$

Quadratic indefinite programs can be efficiently solved with a branch-and-bound approach by means of a suitable eigenvectors-based decomposition of the objective function (see, e.g., Cambini and Sodini 2005, 2008; Fampa et al. 2017). Specifically speaking, since $\hat{Q}$ is a symmetric matrix, there exists an orthonormal matrix $U \in \mathbb{R}^{n \times n}$ ($UU^T = U^T U = I$) and a diagonal matrix $D \in \mathbb{R}^{n \times n}$ such that $\hat{Q} = UDU^T$. The diagonal elements of D are the eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ of $\hat{Q}$, while the orthonormal columns $u_1, \ldots, u_n \in \mathbb{R}^n$ of U are the corresponding eigenvectors of $\hat{Q}$. As a consequence, it results:

$$x^T \hat{Q} x = \sum_{i=1}^{n} \lambda_i (u_i^T x)^2 .$$

Thus, by means of the sets of indices

$$\Lambda^+ = \left\{ i = 1, \ldots, n \;:\; \lambda_i > 0 \right\} \quad , \quad \Lambda^- = \left\{ i = 1, \ldots, n \;:\; \lambda_i < 0 \right\}$$

and the vectors

$$v_i = \sqrt{|\, \lambda_i \,|} \cdot u_i \quad \forall i = 1, \ldots, n \,,$$

the quadratic component of f can be rewritten as follows:

$$x^T \hat{Q} x = \sum_{i \in \Lambda^+} (v_i^T x)^2 - \sum_{i \in \Lambda^-} (v_i^T x)^2 .$$

In this way, f can be expressed in the following *D.C. form*:

$$f(x) = \left(\sum_{i \in \Lambda^+} (v_i^T x)^2 + \hat{a}^T x + \hat{a}_0 \right) - \left(\sum_{i \in \Lambda^-} (v_i^T x)^2 \right) \tag{3}$$

The following branching variables are suggested by (3) for all $i \in \Lambda^-$:

$$\mu_i = v_i^T x$$

so that

$$\underline{\mu}_i = \min_{x \in S} v_i^T x \quad \text{and} \quad \overline{\mu}_i = \max_{x \in S} v_i^T x.$$

Moreover, for all $\mu^L := (\mu_i^L)_{i \in \Lambda^-}$, $\mu^U := (\mu_i^U)_{i \in \Lambda^-} \in \mathbb{R}^{|\Lambda^-|}$ such that $\mu^L \leq \mu^U$, the following rectangle is introduced:

$$\left[\mu^L, \mu^U\right] = \left\{ x \in \mathbb{R}^n : \ \mu_i^L \leq v_i^T x \leq \mu_i^U \quad \forall i \in \Lambda^- \right\}.$$

Let $\underline{\mu} := (\underline{\mu}_i)_{i \in \Lambda^-}$ and $\overline{\mu} := (\overline{\mu}_i)_{i \in \Lambda^-}$. With respect to branching variables μ, rectangle $\left[\underline{\mu}, \overline{\mu}\right]$ is the smallest partition Π_0 containing S. The following underestimation function can then be stated by means of (3).

Theorem 1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be expressed as in (3). Then, the following convex quadratic function is an underestimation function for f over $S \cap \left[\mu^L, \mu^U\right]$:*

$$(u_0) \begin{cases} \Phi_0(x) &= \left(\sum_{i \in \Lambda^+} (v_i^T x)^2 + \hat{a}^T x + \hat{a}_0 \right) + \sum_{i \in \Lambda^-} \left(\mu_i^L \mu_i^U - (\mu_i^L + \mu_i^U) v_i^T x \right), \\ Er_0(x) &= \sum_{i \in \Lambda^-} \left(v_i^T x - \mu_i^L \right) \left(\mu_i^U - v_i^T x \right), \end{cases}$$

with $\frac{1}{4} \sum_{i \in \Lambda^-} (\mu_i^U - \mu_i^L)^2$ the maximum error for $Er_0(x)$ obtained at $v_i^T x = \frac{\mu_i^L + \mu_i^U}{2}$, $i \in \Lambda^-$.

Proof Being $\mu_i^L \leq v_i^T x \leq \mu_i^U$ for all $x \in S \cap \left[\mu^L, \mu^U\right]$ and for all $i \in \Lambda^-$, from Lemma 3 it yields:

$$-(v_i^T x)^2 \geq -(\mu_i^L + \mu_i^U) v_i^T x + \mu_i^L \mu_i^U$$

Hence, $\Phi_0(x)$ follows trivially from (3). Moreover, it results:

$$\begin{aligned} Er_0(x) &= f(x) - \Phi_0(x) \\ &= \sum_{i \in \Lambda^-} \left(-(v_i^T x)^2 + (\mu_i^L + \mu_i^U) v_i^T x - \mu_i^L \mu_i^U \right) \\ &= \sum_{i \in \Lambda^-} \left(v_i^T x - \mu_i^L \right) \left(\mu_i^U - v_i^T x \right) \end{aligned}$$

□

2.5 Splitting process

Assume to be in the iterative process and that the partition having the smallest lower bound LB have been extracted from PQ by means of the command "$[\Pi, opt] := PQextract()$". It is now necessary to choose a branching variable in order to split the partition Π. Assume, for example, that underestimation (u_0) is used (results are analogous for all underestimations functions which will be proposed in the next Section), so that $\Pi = \left[\mu^L, \mu^U\right]$ and:

$$Er_0(x) = \sum_{i \in \Lambda^-} \left(v_i^T x - \mu_i^L\right)\left(\mu_i^U - v_i^T x\right),$$

with maximum error value $\frac{1}{4}\sum_{i\in\Lambda^-}(\mu_i^U - \mu_i^L)^2$ obtained at $v_i^T x = \frac{\mu_i^L + \mu_i^U}{2}$, $i \in \Lambda^-$. Two criteria are generally used to determine the branching variable:

- *The largest interval* $\left[\mu_i^L, \mu_i^U\right]$: this criterion is aimed to reduce as much as possible the error maximum value; on the other hand, it does not use the feasible point $opt = \arg\min_{x\in S\cap\Pi} \Phi_0(x)$ which may suggest where to look for the optimal solution of the problem.
- *The largest error addend* $\left(v_i^T x - \mu_i^L\right)\left(\mu_i^U - v_i^T x\right)$: this criterion is aimed to reduce as much as possible the error value at the point opt, hence tightening the underestimation as much as possible close to opt.

Once the branching variable is chosen, the question is how to split the corresponding interval $\left[\mu_i^L, \mu_i^U\right]$, that is, how to determine the value μ_i^* so that $\left[\mu_i^L, \mu_i^U\right] = \left[\mu_i^L, \mu_i^*\right] \cup \left[\mu_i^*, \mu_i^U\right]$. In this light, possible choices are:

- *The medium point of the interval* $\left[\mu_i^L, \mu_i^U\right]$: the use of $\mu_i^M = \frac{\mu_i^L + \mu_i^U}{2}$ is aimed to reduce as much as possible the error maximum value, again the feasible point opt is not used;
- *The value* $\mu_i^{opt} = v_i^T opt$: in this case the optimal solution of the underestimation is used, but this value may be close to the boundaries μ_i^L and μ_i^U thus highly increasing the number of iterations needed to solve the problem and hence increasing the convergence time itself;
- *A linear combination* of μ_i^M and μ_i^{opt}: given a value $\alpha \in [0, 1]$ the point $\mu_i^* = \alpha\mu_i^{opt} + (1-\alpha)\mu_i^M$ could be used to take into account of both the error maximum value and the solution opt.

These criteria can be summarized and linked as follows:

- Blindly reduce as much as possible the error maximum value: choose the largest interval $\left[\mu_i^L, \mu_i^U\right]$ and split it in the middle with μ_i^M (see, e.g., Cambini and Sodini 2005; Shen et al. 2020);

- Use the "infos" given by *opt*: choose the largest error addend $\left(v_i^T x - \mu_i^L\right)\left(\mu_i^U - v_i^T x\right)$ and split it with respect to $\mu_i^* = \alpha\mu_i^{opt} + (1-\alpha)\mu_i^M$ (see, e.g., Cambini and Salvi 2009, 2010; Fampa et al. 2017).

The recent literature shows that the latter opportunity is the most performing one, and is the one that will be used in the computational test described in Sect. 4. Finally, notice that, by means of Lemma 2, the splitting process should be performed only if the largest error addend is grater than a suitable tolerance $errTol > 0$.

3 Specific underestimation functions

The eigenvectors-based approach described in Sect. 2.4 actually has some drawbacks. First of all, the eigen-decomposition of a quadratic function is an heavy task from both a computational and a numerical point of view. Moreover, such an eigen-decomposition does not take into account of the particular structure of linear multiplicative functions. In this very light, in the recent literature various papers aimed to approach linear multiplicative problems without the use of eigen-decompositions (see, e.g., Shen et al. 2020, 2022; Wang et al. 2012; Zhou et al. 2015).

The aim of this section is to state some underestimation functions for f not using the eigenvectors of matrix $\hat{Q}$, in order to efficiently solve Problem P without the computational and numerical troubles of eigenvectors computing.

3.1 Linear underestimation functions

In the recent literature (see Shen et al. 2020, for example) some linear underestimation functions have been used to solve linear multiplicative problems by means of a branch-and-bound approach. Usually these underestimations are not tight, for this very reason in this subsection some further linear underestimations will be studied. Recalling that:

$$f(x) = \sum_{i=1}^{p} (c_i^T x)(d_i^T x) + \hat{a}^T x + \hat{a}_0,$$

the following $2p$ branching variables can be considered:

$$\xi_i = c_i^T x \ \text{ and } \ \delta_i = d_i^T x\,, \quad i = 1, \ldots, p$$

so that

$$\underline{\xi}_i = \min_{x \in S} c_i^T x\,,\ \overline{\xi}_i = \max_{x \in S} c_i^T x \quad \text{and} \quad \underline{\delta}_i = \min_{x \in S} d_i^T x\,,\ \overline{\delta}_i = \max_{x \in S} d_i^T x\,.$$

Moreover, for all $\xi^L := (\xi_i^L)_{i=1,\dots,p}$, $\xi^U := (\xi_i^U)_{i=1,\dots,p} \in \mathbb{R}^p$ such that $\xi^L \leqq \xi^U$ and for all $\delta^L := (\delta_i^L)_{i=1,\dots,p}$, $\delta^U := (\delta_i^U)_{i=1,\dots,p} \in \mathbb{R}^p$ such that $\delta^L \leqq \delta^U$, the following rectangles are introduced:

$$\begin{aligned}\left[\xi^L, \xi^U\right] &= \left\{x \in \mathbb{R}^n : \ \xi_i^L \le c_i^T x \le \xi_i^U \quad \forall i = 1, \dots, p\right\};\\ \left[\delta^L, \delta^U\right] &= \left\{x \in \mathbb{R}^n : \ \delta_i^L \le d_i^T x \le \delta_i^U \quad \forall i = 1, \dots, p\right\}.\end{aligned}$$

Let $\underline{\xi} := (\underline{\xi}_i)_{i=1,\dots,p}$, $\overline{\xi} := (\overline{\xi}_i)_{i=1,\dots,p}$, $\underline{\delta} := (\underline{\delta}_i)_{i=1,\dots,p}$, and $\overline{\delta} := (\overline{\delta}_i)_{i=1,\dots,p}$. With respect to branching variables ξ and δ, rectangle $\left[\underline{\xi}, \overline{\xi}\right] \cap \left[\underline{\delta}, \overline{\delta}\right]$ is the smallest partition Π_0 containing S.

The following *linear* underestimation functions can then be stated.

Theorem 2 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be expressed as in (2). Then, the following linear functions are underestimation functions for f over $S \cap \left[\xi^L, \xi^U\right] \cap \left[\delta^L, \delta^U\right]$:*

$$(u_1)\begin{cases}\Phi_1(x) &= \left(\hat{a}^T x + \hat{a}_0\right) + \sum_{i=1}^{p}\left(\delta_i^L c_i^T x + \xi_i^L d_i^T x - \xi_i^L \delta_i^L\right),\\ Er_1(x) &= \sum_{i=1}^{p}\left(c_i^T x - \xi_i^L\right)\left(d_i^T x - \delta_i^L\right),\end{cases}$$

and

$$(u_2)\begin{cases}\Phi_2(x) &= \left(\hat{a}^T x + \hat{a}_0\right) + \sum_{i=1}^{p}\left(\delta_i^U c_i^T x + \xi_i^U d_i^T x - \xi_i^U \delta_i^U\right),\\ Er_2(x) &= \sum_{i=1}^{p}\left(\xi_i^U - c_i^T x\right)\left(\delta_i^U - d_i^T x\right),\end{cases}$$

with $\sum_{i=1}^{p}(\xi_i^U - \xi_i^L)(\delta_i^U - \delta_i^L)$ the maximum error for $Er_1(x)$ and $Er_2(x)$ obtained at $c_i^T x = \xi_i^U$, $d_i^T x = \delta_i^U$, $i = 1, \dots, p$, and $c_i^T x = \xi_i^L$, $d_i^T x = \delta_i^L$, $i = 1, \dots, p$, respectively.

Proof For all $i = 1, \dots, p$, $\left(c_i^T x - \xi_i^L\right) \ge 0$, $\left(d_i^T x - \delta_i^L\right) \ge 0$, $\left(\xi_i^U - c_i^T x\right) \ge 0$ and $\left(\delta_i^U - d_i^T x\right) \ge 0$, yield:

$$\begin{aligned}0 \le \left(c_i^T x - \xi_i^L\right)\left(d_i^T x - \delta_i^L\right) &= (c_i^T x)(d_i^T x) - \delta_i^L c_i^T x - \xi_i^L d_i^T x + \xi_i^L \delta_i^L\\ &\Rightarrow (c_i^T x)(d_i^T x) \ge \delta_i^L c_i^T x + \xi_i^L d_i^T x - \xi_i^L \delta_i^L,\\ 0 \le \left(\xi_i^U - c_i^T x\right)\left(\delta_i^U - d_i^T x\right) &= (c_i^T x)(d_i^T x) - \delta_i^U c_i^T x - \xi_i^U d_i^T x + \xi_i^U \delta_i^U\\ &\Rightarrow (c_i^T x)(d_i^T x) \ge \delta_i^U c_i^T x + \xi_i^U d_i^T x - \xi_i^U \delta_i^U\end{aligned}$$

Hence, the thesis follows by means of simple calculations. □

Remark 3 Notice that $\delta_i^L c_i^T x + \xi_i^L d_i^T x - \xi_i^L \delta_i^L$ and $\delta_i^U c_i^T x + \xi_i^U d_i^T x - \xi_i^U \delta_i^U$ are know as the McCormick lower envelopes for the bilinear function $(c_i^T x)(d_i^T x)$ (McCormick

1976). Notice also that in the case $\xi^L \geq 0$ and $\delta^L \geq 0$, $i = 1, \ldots, p$, linear underestimations of the kind $\sum_{i=1}^{p}(c_i^T x)\delta_i^L + \hat{a}^T x + \hat{a}_0$ or $\sum_{i=1}^{p}(d_i^T x)\xi_i^L + \hat{a}^T x + \hat{a}_0$ have been used, and that these are far less tight than the one proposed in this subsection.

3.2 Quadratic underestimation functions

The aim of this subsection is to state underestimation functions of f, tighter than the linear ones, by properly rewriting f in D.C. form.

Theorem 3 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be defined as in (2). Then, f can be rewritten in the following D.C. forms*:

$$\text{(i)} \quad f(x) = \left(\frac{1}{2}\sum_{i=1}^{p}(c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0\right) - \left(\frac{1}{2}\sum_{i=1}^{p}\left((c_i^T x)^2 + (d_i^T x)^2\right)\right);$$

$$\text{(ii)} \quad f(x) = \left(\frac{1}{4}\sum_{i=1}^{p}(c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0\right) - \left(\frac{1}{4}\sum_{i=1}^{p}(c_i^T x - d_i^T x)^2\right);$$

$$\text{(iii)} \quad f(x) = \left(\frac{1}{2}\sum_{i=1}^{p}\left((c_i^T x)^2 + (d_i^T x)^2\right) + \hat{a}^T x + \hat{a}_0\right) - \left(\frac{1}{2}\sum_{i=1}^{p}(c_i^T x - d_i^T x)^2\right).$$

Proof Firstly, one get:

$$\triangleright \quad (c_i^T x)(d_i^T x) = \frac{1}{2}\left(2(c_i^T x)(d_i^T x) + (c_i^T x)^2 + (d_i^T x)^2 - (c_i^T x)^2 - (d_i^T x)^2\right)$$
$$= \frac{1}{2}\left((c_i^T x + d_i^T x)^2 - (c_i^T x)^2 - (d_i^T x)^2\right);$$

$$\triangleright \quad (c_i^T x)(d_i^T x) = \frac{1}{4}\left(2(c_i^T x)(d_i^T x) + (c_i^T x)^2 + (d_i^T x)^2 + 2(c_i^T x)(d_i^T x) - (c_i^T x)^2 - (d_i^T x)^2\right)$$
$$= \frac{1}{4}\left((c_i^T x + d_i^T x)^2 - (c_i^T x - d_i^T x)^2\right);$$

$$\triangleright \quad (c_i^T x)(d_i^T x) = \frac{1}{2}\left((c_i^T x)^2 + (d_i^T x)^2 + 2(c_i^T x)(d_i^T x) - (c_i^T x)^2 - (d_i^T x)^2\right)$$
$$= \frac{1}{2}\left((c_i^T x)^2 + (d_i^T x)^2 - (c_i^T x - d_i^T x)^2\right).$$

Then, by opportunely replacing each of them in (2), the thesis follows. □

The following underestimation function can be stated by means of *(i)* of Theorem 3 and the $2p$ branching variables described in the previous subsection.

Theorem 4 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be expressed as in (i) of Theorem 3. Then, the following convex quadratic function is an underestimation function for f over $S \cap \left[\xi^L, \xi^U\right] \cap \left[\delta^L, \delta^U\right]$:*

$$(u_3)\begin{cases}\Phi_3(x) &= \left(\frac{1}{2}\sum_{i=1}^{p}(c_i^Tx+d_i^Tx)^2+\hat{a}^Tx+\hat{a}_0\right)\\ &\quad+\frac{1}{2}\sum_{i=1}^{p}\left(\xi_i^L\xi_i^U+\delta_i^L\delta_i^U-(\xi_i^L+\xi_i^U)c_i^Tx-(\delta_i^L+\delta_i^U)d_i^Tx\right),\\ Er_3(x) &= \frac{1}{2}\sum_{i=1}^{p}\left((c_i^Tx-\xi_i^L)(\xi_i^U-c_i^Tx)+(d_i^Tx-\delta_i^L)(\delta_i^U-d_i^Tx)\right),\end{cases}$$

with $\frac{1}{8}\sum_{i=1}^{p}(\xi_i^U-\xi_i^L)^2+\frac{1}{8}\sum_{i=1}^{p}(\delta_i^U-\delta_i^L)^2$ *the maximum error for* $Er_3(x)$ *obtained at* $c_i^Tx=\frac{\xi_i^L+\xi_i^U}{2}$ *and* $d_i^Tx=\frac{\delta_i^L+\delta_i^U}{2}$, $i=1,\dots,p$.

Proof Being $\xi_i^L\le c_i^Tx\le\xi_i^U$ and $\delta_i^L\le d_i^Tx\le\delta_i^U$ for all $x\in S\cap\left[\xi^L,\xi^U\right]\cap\left[\delta^L,\delta^U\right]$ and for all $i=1,\dots,p$, from Lemma 3 it yields:

$$-(c_i^Tx)^2-(d_i^Tx)^2\ge-(\xi_i^L+\xi_i^U)c_i^Tx+\xi_i^L\xi_i^U-(\delta_i^L+d_i^U)d_i^Tx+\delta_i^L\delta_i^U$$

Hence, $\Phi_3(x)$ follows trivially from *(i)* of Theorem 3. Moreover, it results:

$$\begin{aligned}Er_3(x) =& f(x)-\Phi_3(x)\\ =&\frac{1}{2}\sum_{i=1}^{p}\left(2(c_i^Tx)(d_i^Tx)-(c_i^Tx+d_i^Tx)^2+(\xi_i^L+\xi_i^U)c_i^Tx\right.\\ &\left.+(\delta_i^L+\delta_i^U)d_i^Tx-(\xi_i^L\xi_i^U+\delta_i^L\delta_i^U)\right)\\ =&\frac{1}{2}\sum_{i=1}^{p}\left(-(c_i^Tx)^2-(d_i^Tx)^2+(\xi_i^L+\xi_i^U)c_i^Tx\right.\\ &\left.+(\delta_i^L+\delta_i^U)d_i^Tx-(\xi_i^L\xi_i^U+\delta_i^L\delta_i^U)\right)\\ =&\frac{1}{2}\sum_{i=1}^{p}\left((c_i^Tx-\xi_i^L)(\xi_i^U-c_i^Tx)+(d_i^Tx-\delta_i^L)(\delta_i^U-d_i^Tx)\right)\end{aligned}$$

□

In similar way, for all $i=1,\dots,p$, the following branching variables are suggested by *(ii)* and *(iii)* of Theorem 3:

$$\sigma_i=(c_i-d_i)^Tx$$

so that

$$\underline{\sigma}_i=\min_{x\in S}(c_i-d_i)^Tx\quad\text{and}\quad\overline{\sigma}_i=\max_{x\in S}(c_i-d_i)^Tx$$

Moreover, for all $\sigma^L:=(\sigma_i^L)_{i=1,\dots,p}$, $\sigma^L:=(\sigma_i^L)_{i=1,\dots,p}\in\mathbb{R}^p$ such that $\sigma^L\le\sigma^U$, the following rectangle is introduced:

$$\left[\sigma^L,\sigma^U\right]=\left\{x\in\mathbb{R}^n\ :\ \sigma_i^L\le(c_i-d_i)^Tx\le\sigma_i^U\quad\forall i=1,\dots,p\right\}.$$

Let $\underline{\sigma} := (\underline{\sigma}_i)_{i=1,\dots,p}$ and $\overline{\sigma} := (\overline{\sigma}_i)_{i=1,\dots,p}$. With respect to branching variables σ, rectangle $\left[\underline{\sigma}, \overline{\sigma}\right]$ is the smallest partition Π_0 containing S. In this light, the following underestimation functions can be stated by means of *(ii)* and *(iii)* of Theorem 3.

Theorem 5 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be expressed as in (ii) and i(ii) of Theorem 3. Then, the following convex quadratic functions are underestimation functions for f over $S \cap \left[\sigma^L, \sigma^U\right]$, respectively*:

$$(u_4)\begin{cases} \Phi_4(x) &= \left(\frac{1}{4}\sum_{i=1}^{p}(c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0\right) \\ &\quad + \frac{1}{4}\sum_{i=1}^{p}\left(\sigma_i^L\sigma_i^U - (\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x)\right), \\ Er_4(x) &= \frac{1}{4}\sum_{i=1}^{p}\left((c_i^T x - d_i^T x) - \sigma_i^L\right)\left(\sigma_i^U - (c_i^T x - d_i^T x)\right), \end{cases}$$

and

$$(u_5)\begin{cases} \Phi_5(x) &= \left(\frac{1}{2}\sum_{i=1}^{p}\left((c_i^T x)^2 + (d_i^T x)^2\right) + \hat{a}^T x + \hat{a}_0\right) \\ &\quad + \frac{1}{2}\sum_{i=1}^{p}\left(\sigma_i^L\sigma_i^U - (\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x)\right), \\ Er_5(x) &= \frac{1}{4}\sum_{i=1}^{p}\left((c_i^T x - d_i^T x) - \sigma_i^L\right)\left(\sigma_i^U - (c_i^T x - d_i^T x)\right), \end{cases}$$

with $\frac{1}{16}\sum_{i=1}^{p}(\sigma_i^U - \sigma_i^L)^2$ and $\frac{1}{8}\sum_{i=1}^{p}(\sigma_i^U - \sigma_i^L)^2$ the maximum error for $Er_4(x)$ and $Er_5(x)$, respectively, obtained at $(c_i^T x - d_i^T x) = \frac{\sigma_i^L + \sigma_i^U}{2}$, $i = 1, \dots, p$.

Proof Being $\sigma_i^L \le (c_i^T x - d_i^T x) \le \sigma_i^U$ for all $x \in S \cap \left[\sigma^L, \sigma^U\right]$ and for all $i = 1, \dots, p$, from Lemma 3 it yields:

$$-(c_i^T x - d_i^T x)^2 \ge -(\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x) + \sigma_i^L\sigma_i^U$$

Hence, $\Phi_4(x)$ and $\Phi_5(x)$ follow, respectively, from *(ii)* and *(iii)* of Theorem 3. Moreover, it results:

$$\begin{aligned} Er_4(x) =& f(x) - \Phi_4(x) \\ &= \frac{1}{4}\sum_{i=1}^{p}\left(4(c_i^T x)(d_i^T x) - (c_i^T x + d_i^T x)^2 + (\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x) - \sigma_i^L\sigma_i^U\right) \\ &= \frac{1}{4}\sum_{i=1}^{p}\left(-(c_i^T x - d_i^T x)^2 + (\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x) - \sigma_i^L\sigma_i^U\right) \\ &= \frac{1}{4}\sum_{i=1}^{p}\left((c_i^T x - d_i^T x) - \sigma_i^L\right)\left(\sigma_i^U - (c_i^T x - d_i^T x)\right) \end{aligned}$$

and

$$\begin{aligned} Er_5(x) &= f(x) - \Phi_5(x) \\ &= \frac{1}{2}\sum_{i=1}^{p}\left(2(c_i^T x)(d_i^T x) - (c_i^T x)^2 - (d_i^T x)^2 + (\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x) - \sigma_i^L \sigma_i^U\right) \\ &= \frac{1}{2}\sum_{i=1}^{p}\left(-(c_i^T x - d_i^T x)^2 + (\sigma_i^L + \sigma_i^U)(c_i^T x - d_i^T x) - \sigma_i^L \sigma_i^U\right) \\ &= \frac{1}{2}\sum_{i=1}^{p}\left((c_i^T x - d_i^T x) - \sigma_i^L\right)\left(\sigma_i^U - (c_i^T x - d_i^T x)\right) \end{aligned}$$

□

Remark 4 Notice that (u_4) dominates (u_5) since $Er_5(x) = 2 \cdot Er_4(x)$. For this very reason, (u_5) has been given just for the sake of completeness and will no more be used in the rest of the paper.

3.3 Further hybrid underestimation functions

For the sake of completeness, some more underestimation functions of f will be studied by applying the eigendecomposition approach to the D.C. forms provided by *(i)* and *(ii)* of Theorem 3. Specifically speaking, each of them can be rewritten as follows:

$$\text{(i)} \quad f(x) = \left(\frac{1}{2}\sum_{i=1}^{p}(c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0\right) - \left(x^T Q_1 x\right),$$

$$\text{(ii)} \quad f(x) = \left(\frac{1}{4}\sum_{i=1}^{p}(c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0\right) - \left(x^T Q_2 x\right),$$

with $Q_1 = \frac{1}{2}\sum_{i=1}^{p}(c_i c_i^T + d_i d_i^T)$ and $Q_2 = \frac{1}{4}\sum_{i=1}^{p}(c_i - d_i)(c_i - d_i)^T$ symmetric positive semidefinite matrices. Hence, there exist two orthonormal matrices $\tilde{U}, \hat{U} \in \mathbb{R}^{n\times n}$ and two diagonal matrices $\tilde{D}, \hat{D} \in \mathbb{R}^{n\times n}$ such that $Q_1 = \tilde{U}\tilde{D}\tilde{U}^T$ and $Q_2 = \hat{U}\hat{D}\hat{U}^T$. The diagonal elements of $\tilde{D}$ are the nonnegative eigenvalues $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n \in \mathbb{R}$ of Q_1, while the orthonormal columns $\tilde{u}_1, \ldots, \tilde{u}_n \in \mathbb{R}^n$ of $\tilde{U}$ are the corresponding eigenvectors of Q_1; in similar way, the diagonal elements of $\hat{D}$ are the nonnegative eigenvalues $\hat{\lambda}_1, \ldots, \hat{\lambda}_n \in \mathbb{R}$ of Q_2, while the orthonormal columns $\hat{u}_1, \ldots, \hat{u}_n \in \mathbb{R}^n$ of $\hat{U}$ are the corresponding eigenvectors of Q_2. As a consequence, since Q_1 and Q_2 have no negative eigenvalues, it results:

$$x^T Q_1 x = \sum_{i=1}^{n}\tilde{\lambda}_i(\tilde{u}_i^T x)^2 = \sum_{i\in\Theta^+}(\tilde{v}_i^T x)^2 \quad \text{and} \quad x^T Q_2 x = \sum_{i=1}^{n}\hat{\lambda}_i(\hat{u}_i^T x)^2 = \sum_{i\in\Gamma^+}(\hat{v}_i^T x)^2$$

with $\Theta^+ = \{i = 1, \dots, n \ : \ \tilde{\lambda}_i > 0\}$, $\Gamma^+ = \{i = 1, \dots, n \ : \ \hat{\lambda}_i > 0\}$, $\tilde{v}_i = \sqrt{\tilde{\lambda}_i} \cdot \tilde{u}_i$ for all $i \in \Theta^+$, and $\hat{v}_i = \sqrt{\hat{\lambda}_i} \cdot \hat{u}_i$ for all $\forall i \in \Gamma^+$. Hence, the following further D.C. forms hold:

$$f(x) = \left(\frac{1}{2} \sum_{i=1}^{p} (c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0 \right) - \left(\sum_{i \in \Theta^+} (\tilde{v}_i^T x)^2 \right); \tag{4}$$

$$f(x) = \left(\frac{1}{4} \sum_{i=1}^{p} (c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0 \right) - \left(\sum_{i \in \Gamma^+} (\hat{v}_i^T x)^2 \right). \tag{5}$$

In this light, the following branching variables are suggested by (4) and (5), respectively:

$$\theta_i = \tilde{v}_i^T x \, , \ i \in \Theta^+ \quad \text{and} \quad \gamma_i = \hat{v}_i^T x \, , \ i \in \Gamma^+$$

so that

$$\underline{\theta}_i = \min_{x \in S} \tilde{v}_i^T x \, , \ \overline{\theta}_i = \max_{x \in S} \tilde{v}_i^T x \quad \text{and} \quad \underline{\gamma}_i = \min_{x \in S} \hat{v}_i^T x \, , \ \overline{\gamma}_i = \max_{x \in S} \hat{v}_i^T x$$

For the sake of convenience, let $\underline{\theta} := (\underline{\theta}_i)_{i \in \Theta^+}$, $\overline{\theta} := (\overline{\theta}_i)_{i \in \Theta^+}$, $\underline{\gamma} := (\underline{\gamma}_i)_{i \in \Gamma^+}$, and $\overline{\gamma} := (\overline{\gamma}_i)_{i \in \Gamma^+}$. Moreover, for all $\theta^L := (\theta_i^L)_{i \in \Theta^+}$ and $\theta^U := (\theta_i^U)_{i \in \Theta^+}$ such that $\theta^L \leqq \theta^U$ and for all $\gamma^L := (\gamma_i^L)_{i \in \Gamma^+}$ and $\gamma^U := (\gamma_i^U)_{i \in \Gamma^+}$ such that $\gamma^L \leqq \gamma^U$, the following rectangles are introduced:

$$\begin{aligned} \left[\theta^L, \theta^U\right] &= \left\{ x \in \mathbb{R}^n \ : \ \theta_i^L \leq \tilde{v}_i^T x \leq \theta_i^U \quad \forall i \in \Theta^+ \right\}; \\ \left[\gamma^L, \gamma^U\right] &= \left\{ x \in \mathbb{R}^n \ : \ \gamma_i^L \leq \hat{v}_i^T x \leq \gamma_i^U \quad \forall i \in \Gamma^+ \right\}. \end{aligned}$$

Rectangles $\left[\underline{\theta}, \overline{\theta}\right]$ and $\left[\underline{\gamma}, \overline{\gamma}\right]$ result to be the smallest partitions Π_0 containing S with respect to branching variables θ and γ, respectively.

Theorem 6 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be expressed as in (i) and (ii) of Theorem 3. Then, the following convex quadratic functions are underestimation functions for f over $S \cap \left[\theta^L, \theta^U\right]$ and $S \cap \left[\gamma^L, \gamma^U\right]$, respectively*:

$$(u_6) \begin{cases} \Phi_6(x) \ = \left(\frac{1}{2} \sum_{i=1}^{p} (c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0 \right) \\ \qquad\qquad + \sum_{i \in \Theta^+} \left(\theta_i^L \theta_i^U - (\theta_i^L + \theta_i^U) \tilde{v}_i^T x \right), \\ Er_6(x) \ = \sum_{i \in \Theta^+} \left(\tilde{v}_i^T x - \theta_i^L \right) \left(\theta_i^U - \tilde{v}_i^T x \right), \end{cases}$$

and

$$(u_7)\begin{cases}\Phi_7(x) &= \left(\frac{1}{4}\sum_{i=1}^{p}(c_i^T x + d_i^T x)^2 + \hat{a}^T x + \hat{a}_0\right) \\ &\quad + \sum_{i\in\Gamma^+}\left(\gamma_i^L\gamma_i^U - (\gamma_i^L + \gamma_i^U)\hat{v}_i^T x\right), \\ Er_7(x) &= \sum_{i\in\Gamma^+}\left(\hat{v}_i^T x - \gamma_i^L\right)\left(\gamma_i^U - \hat{v}_i^T x\right),\end{cases}$$

with $\frac{1}{4}\sum_{i\in\Theta^+}(\theta_i^U - \theta_i^L)^2$ *and* $\frac{1}{4}\sum_{i\in\Gamma^+}(\gamma_i^U - \gamma_i^L)^2$ *the maximum errors for* $Er_6(x)$ *and* $Er_7(x)$, *respectively*, *obtained at* $\tilde{v}_i^T x = \frac{\theta_i^L + \theta_i^U}{2}$, $i \in \Theta^+$, *and* $\hat{v}_i^T x = \frac{\gamma_i^L + \gamma_i^U}{2}$, $i \in \Gamma^+$.

Proof The thesis follows in the same lines of Theorems 4 and 5. □

3.4 A particular case

In many applicative problems (see the forthcoming Sect. 5), the linear multiplicative objective function $f : \mathbb{R}^n \to \mathbb{R}$ has the following particular structure:

$$f(z, \beta, g) = \sum_{i=1}^{p}\beta_i g_i + q(z, \beta, g), \tag{6}$$

where $z \in \mathbb{R}^m$, $\beta := (\beta_i)_{i=1,\ldots,p} \in \mathbb{R}^p$, $g := (g_i)_{i=1,\ldots,p} \in \mathbb{R}^p$, $q(z, \beta, g)$ a linear or a convex quadratic term, and $n := m + 2p$. In this light, it is worth studying the behavior of the underestimation functions proposed so far in the particular case of objective functions of type (6).

Theorem 7 *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be expressed as in* (6). *Then*, *the following underestimation functions for f are stated*:

(i) *for* $\beta \in [\beta^L, \beta^U]$ *and* $g \in [g^L, g^U]$, *it results*

$$\Phi_1(z, \beta, g) = q(z, \beta, g) + \sum_{i=1}^{p}\left(g_i^L\beta_i + \beta_i^L g_i - \beta_i^L g_i^L\right),$$

$$\Phi_2(z, \beta, g) = q(z, \beta, g) + \sum_{i=1}^{p}\left(g_i^U\beta_i + \beta_i^U g_i - \beta_i^U g_i^U\right),$$

$$\Phi_3(z, \beta, g) = \left(\frac{1}{2}\sum_{i=1}^{p}(\beta_i + g_i)^2 + q(z, \beta, g)\right) + \frac{1}{2}\sum_{i=1}^{p}\left(\beta_i^L\beta_i^U + g_i^L g_i^U - (\beta_i^L + \beta_i^U)\beta_i - (g_i^L + g_i^U)g_i\right),$$

and $\Phi_6(z, \beta, g) = \Phi_3(z, \beta, g)$;

(ii) *for* $(\beta - g) \in [\sigma^L, \sigma^U]$, *it results*

$$\Phi_4(z,\beta,g) = \left(\frac{1}{4}\sum_{i=1}^{p}(\beta_i+g_i)^2 + q(z,\beta,g)\right) + \frac{1}{4}\sum_{i=1}^{p}\left(\sigma_i^L\sigma_i^U - (\sigma_i^L+\sigma_i^U)(\beta_i-g_i)\right)$$

and $\Phi_0(z,\beta,g) = \Phi_7(z,\beta,g) = \Phi_4(z,\beta,g)$.

Proof Firstly, from Theorem 3, it results:

$$\triangleright\ f(z,\beta,g) = \left(\frac{1}{2}\sum_{i=1}^{p}(\beta_i+g_i)^2 + q(z,\beta,g)\right) - \left(\frac{1}{2}\sum_{i=1}^{p}\left(\beta_i^2+g_i^2\right)\right),$$

$$\triangleright\ f(z,\beta,g) = \left(\frac{1}{4}\sum_{i=1}^{p}(\beta_i+g_i)^2 + q(z,\beta,g)\right) - \left(\frac{1}{4}\sum_{i=1}^{p}(\beta_i-g_i)^2\right)$$

Hence, the introduced Φ_1, Φ_2, Φ_3 and Φ_4 are underestimation functions for f due to Theorems 3, 4 and 5. As regards to Φ_0 notice that:

$$\beta_i g_i = \frac{1}{2}(\beta_i,g_i)\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{pmatrix}\beta_i\\ g_i\end{pmatrix}.$$

In particular, $\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ has eigenvalues $\lambda = 1$ and $\lambda = -1$ with corresponding eigenvectors $(1,1)^T$ and $(1,-1)^T$, respectively; hence, $\left((1,-1)(\beta_i,g_i)^T\right)^2 = (\beta_i-g_i)^2$. Instead, with respect to Φ_6, notice that:

$$\beta_i^2+g_i^2 = (\beta_i,g_i)\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{pmatrix}\beta_i\\ g_i\end{pmatrix}.$$

Being that $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ has eigenvalue $\lambda = 1$ with algebraic multiplicity equal to 2 and orthogonal eigenvectors $(1,0)^T$ and $(0,1)^T$, then $\left((1,0)(\beta_i,g_i)^T\right)^2 + \left((0,1)(\beta_i,g_i)^T\right)^2 = \beta_i^2+g_i^2$. Furthermore, relatively to Φ_7, notice that:

$$(\beta_i-g_i)^2 = (\beta_i,g_i)\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{pmatrix}\beta_i\\ g_i\end{pmatrix}.$$

In particular, $\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$ has eigenvalues $\lambda = 2$ and $\lambda = 0$ with corresponding eigenvectors $(1,-1)^T$ and $(1,1)^T$, respectively; hence, $\left((1,-1)(\beta_i,g_i)^T\right)^2 = (\beta_i-g_i)^2$. □

Remark 5 It is worth to underlay that *(ii)* of Theorem 7 results to be of great interest in the light of the computational results that will be presented in the next Section.

4 A computational experience

The solution method described and discussed in the previous sections has been implemented in a macOS 12.5.1 environment with an M1 Pro 10-core processor, MATLAB 2022a for coding and Gurobi 9.5.2 as solver for LP and QP problems. In this section, the results of some computational tests are presented, where the performances are compared with respect to the various underestimation functions previously studied. In this light, various instances have been randomly generated by using the "randi()" MATLAB function (integer numbers generated with uniform distribution). The average times spent to solve the instances (obtained with the "tic" and "toc" MATLAB commands), as well as the average number of iterations in procedure "*Solve*()" needed to solve them, are given as results of the computational tests. The used tolerance parameters are $relTol = 2^{-35}$ and $errTol = 2^{-20}$. For the sake of simplicity, in the instances generation we fixed the values $a_0 = 0$ and $c_{0i}, d_{0i} = 0$ for all $i \in 1, \ldots, p$, we considered no equality constraints $A_{eq}x = b_{eq}$ and a number of inequality constraints $m = 2n$. Moreover, the two following cases are taken into account in the instances generation:

- *A "general" case*, where vectors a, , c_i and d_i have been randomly generated with components in the interval $[-4, 4]$, vectors l_b and u_b have been generated with components in the interval $[-10, 10]$, matrix A_{in} has been generated with components in the interval $[-10, 10]$, b_{in} has been generated in order to guarantee a feasible region different from the box $[l_b, u_b]$ and nonempty;
- *A "nonnegative" case*, where vectors c_i and d_i have been randomly generated with components in the interval [0, 4], vector a has been generated with components in the interval $[-4, 4]$, vectors l_b and u_b have been generated with components in the interval [0, 15], matrix A_{in} has been generated with components in the interval $[-10, 10]$, b_{in} has been generated in order to guarantee a feasible region different from the box $[l_b, u_b]$ and nonempty.

Remark 6 The "nonnegative" case is aimed to study the behavior of the underestimation functions when both variables and branching variables are nonnegative, just like sometimes assumed in the literature (see, e.g., Zhou et al. 2015; Shen et al. 2020) and thus covering as a particular case the applicative problems described in Sect. 4.

Table 1 Average number of iterations

		General			Nonneg		
		$n = 10\ p = 4$	$n = 10\ p = 6$	$n = 20\ p = 4$	$n = 10\ p = 4$	$n = 10\ p = 6$	$n = 20\ p = 4$
Eigen	u_0	84.3	147.8	120.0	39.2	47.6	43.9
Lin	u_1	2665.4	10998.0	8540.8	74.9	125.0	98.7
	u_2	2659.6	10838.0	8577.0	95.3	152.7	104.4
Quad	u_3	1448.3	5532.7	2455.2	86.7	143.8	106.9
	u_4	100.8	224.7	132.6	41.4	66.1	44.3
Hybrid	u_6	786.9	1656.3	2100.0	92.0	135.6	120.9
	u_7	95.9	206.0	132.9	41.2	62.9	43.8

4.1 A first comparison of all the underestimations

First of all, it is worth comparing all the introduced underestimation functions. The standard value $\alpha = 0.5$ has been assumed for splitting all the underestimations. Starting from instances having $n = 10$ and $p = 4$, the behaviors with p increased to $p = 6$ and with n increased to $n = 20$ are considered too. Moreover, both the "general" and the "nonnegative" cases are taken into account. The average times and average iterations are given as results of this first computational test and are summarized in the following tables. Six groups of instances (depending on n, p, and "general"/"nonnegative" cases) and 100 instances for each group are considered, with a grand total of 4200 problems solved.

The results provided in Tables 1 and 2 point out that:

- In the "general" case the linear underestimation functions provide very bad tightness and hence very bad performance, while in the "nonnegative" case their performance has the same order of magnitude of the other underestimations;
- Among the quadratic underestimation functions, u_4 is always much better than u_3;
- Among the hybrid underestimation functions, u_7 is always much better than u_6 (this follows being u_6 derived from u_3 and being u_7 derived from u_4);

Table 2 Average elapsed times (secs)

		General			Nonneg		
		$n = 10\ p = 4$	$n = 10\ p = 6$	$n = 20\ p = 4$	$n = 10\ p = 4$	$n = 10\ p = 6$	$n = 20\ p = 4$
Eigen	u_0	0.631	1.155	1.158	0.300	0.359	0.433
Lin	u_1	14.882	61.505	53.822	0.491	0.798	0.699
	u_2	14.556	59.219	51.405	0.603	0.957	0.732
Quad	u_3	11.926	44.801	26.117	0.800	1.451	1.194
	u_4	0.723	1.676	1.224	0.312	0.509	0.397
Hybrid	u_6	5.906	13.244	19.365	0.730	1.149	1.185
	u_7	0.694	1.515	1.206	0.324	0.503	0.442

Table 3 Average number of iterations—"general" case—$n = 25$

	u_4			u_0			u_7		
	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$
$\alpha = 0.0$	195.9	1284.0	7056.1	176.4	1325.3	3811.7	193.3	1265.0	6463.1
$\alpha = 0.1$	186.3	1247.8	6926.6	169.6	988.5	3720.7	187.4	1224.6	6415.9
$\alpha = 0.2$	180.3	1200.8	6794.1	159.2	959.1	**3666.5**	177.6	1187.8	**6338.3**
$\alpha = 0.3$	170.6	1178.9	**6739.0**	153.1	936.3	3684.3	169.9	1156.4	6347.2
$\alpha = 0.4$	162.4	1147.6	6792.6	146.2	913.8	3714.3	163.0	1141.0	6441.1
$\alpha = 0.5$	158.1	1132.0	6805.5	144.2	**900.6**	3802.1	157.6	**1128.9**	6600.1
$\alpha = 0.6$	152.0	**1127.9**	7009.1	136.4	900.9	3978.9	151.6	1138.4	6905.1
$\alpha = 0.7$	149.2	1143.3	7439.6	132.3	918.0	4291.4	149.3	1154.8	7439.3
$\alpha = 0.8$	**145.6**	1191.2	8231.2	130.0	952.2	4832.1	**146.5**	1205.2	8329.9
$\alpha = 0.9$	147.3	1311.6	10015.0	**129.8**	1067.0	5958.8	146.8	1315.1	10129.0
$\alpha = 1.0$	172.7	2135.8	22489.0	155.0	1809.4	14413.0	174.47	2155.2	21709.0

Table 4 Average elapsed times (secs)—"general" case—$n = 25$

	u_4			u_0			u_7		
	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$
$\alpha = 0.0$	1.891	13.000	76.716	1.789	19.810	42.480	1.944	13.064	72.129
$\alpha = 0.1$	1.774	12.363	73.816	1.711	10.197	40.752	1.861	12.534	70.455
$\alpha = 0.2$	1.694	11.792	71.448	1.586	9.774	39.573	1.767	11.990	68.646
$\alpha = 0.3$	1.600	11.495	70.162	1.547	9.440	39.193	1.683	11.537	**67.658**
$\alpha = 0.4$	1.511	11.013	69.728	1.441	9.116	**38.941**	1.601	11.249	67.702
$\alpha = 0.5$	1.465	10.756	**68.820**	1.420	8.864	39.232	1.533	11.014	68.282
$\alpha = 0.6$	1.392	**10.611**	70.073	1.327	**8.775**	40.617	1.470	**10.993**	70.551
$\alpha = 0.7$	1.352	10.646	73.276	1.283	8.853	43.236	1.429	11.051	74.986
$\alpha = 0.8$	**1.304**	10.979	80.001	1.257	9.072	47.891	1.394	11.396	82.436
$\alpha = 0.9$	1.306	11.976	95.771	**1.238**	10.066	58.283	**1.383**	12.330	99.103
$\alpha = 1.0$	1.510	19.516	216.62	1.443	16.996	141.56	1.611	20.099	212.530

- The most performing underestimation functions are always u_0, u_4, u_7;
- In the "nonnegative" case, the performance of the underestimation functions are always better than the "general" case: this is due to the tightness of underestimation functions which results to be more effective in the "nonnegative" case than in the "general" one;
- Among the linear and quadratic underestimation functions, increasing the value of p affects the performances much more than increasing the number of variables n;

Table 5 Average number of iterations—"nonnegative" case—$n = 25$

	u_4			u_0			u_7		
	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$
$\alpha = 0.0$	62.97	126.16	199.00	62.44	118.45	178.02	62.40	123.71	194.68
$\alpha = 0.1$	59.59	122.47	184.76	58.08	111.75	168.91	58.39	118.66	182.34
$\alpha = 0.2$	55.44	112.74	177.26	55.24	107.09	156.77	54.54	109.46	171.16
$\alpha = 0.3$	52.71	107.44	167.05	51.96	99.74	146.06	51.43	103.20	159.63
$\alpha = 0.4$	49.04	101.24	161.72	48.67	95.01	141.23	48.17	97.67	152.05
$\alpha = 0.5$	46.01	96.71	154.76	45.4	89.56	136.96	45.01	91.14	145.47
$\alpha = 0.6$	42.78	91.51	**149.99**	42.74	85.68	130.78	42.35	85.33	137.41
$\alpha = 0.7$	39.86	86.94	152.11	39.89	82.71	**129.86**	39.14	81.44	**135.08**
$\alpha = 0.8$	37.15	**86.55**	166.07	37.29	**82.65**	136.89	36.03	**78.85**	138.32
$\alpha = 0.9$	35.12	96.53	246.28	35.26	92.62	175.11	34.15	83.54	170.98
$\alpha = 1.0$	**27.09**	205.61	1425.20	**26.24**	191.41	1364.00	**26.77**	207.37	1461.30

Table 6 Average elapsed times (secs)—"nonnegative" case—$n = 25$

	u_4			u_0			u_7		
	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$	$p = 4$	$p = 7$	$p = 10$
$\alpha = 0.0$	0.630	1.316	2.179	0.682	1.280	1.953	0.698	1.353	2.182
$\alpha = 0.1$	0.591	1.264	1.997	0.640	1.213	1.855	0.654	1.298	2.041
$\alpha = 0.2$	0.551	1.171	1.905	0.609	1.166	1.717	0.616	1.204	1.915
$\alpha = 0.3$	0.523	1.107	1.789	0.576	1.088	1.603	0.583	1.140	1.783
$\alpha = 0.4$	0.490	1.048	1.731	0.546	1.038	1.553	0.554	1.082	1.706
$\alpha = 0.5$	0.462	0.998	1.649	0.515	0.982	1.506	0.522	1.009	1.629
$\alpha = 0.6$	0.427	0.943	**1.600**	0.487	0.941	1.440	0.496	0.949	1.540
$\alpha = 0.7$	0.400	0.896	1.614	0.459	0.908	**1.430**	0.464	0.910	**1.513**
$\alpha = 0.8$	0.372	**0.889**	1.749	0.433	**0.903**	1.502	0.432	**0.880**	1.542
$\alpha = 0.9$	0.352	0.981	2.561	0.413	0.999	1.893	0.415	0.924	1.873
$\alpha = 1.0$	**0.274**	1.988	14.288	**0.324**	1.938	13.916	**0.341**	2.082	14.727

- Performances follow the number of branching variables of the various underestimation functions; in this light, recall that u_1, u_2 and u_3 have $2p$ branching variables, u_4 has p branching variables, while u_0, u_6 and u_7 have $|\Lambda^-|$, $|\Theta^+|$ and $|\Gamma^+|$ branching variables, respectively.

4.2 A deep comparison of u_0, u_4, u_7—part 1

The previous subsection pointed out that the most performing underestimations are u_0, u_4 and u_7 (and recall that, in the particular case of Sect. 4, these underestimations coincide). The aim of this subsection is to focus on the behavior of

Table 7 Average number of iterations—"general" case—$p = 10$

	u_4			u_0			u_7		
	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$
$\alpha = 0.0$	819.8	1590.0	4966.4	39.0	163.8	1928.6	173.7	1009.6	4295.3
$\alpha = 0.1$	763.6	1530.7	4851.3	35.9	155.9	1879.4	168.2	976.4	4259.1
$\alpha = 0.2$	736.9	1466.3	4788.1	34.3	147.2	1832.6	165.2	933.6	4177.0
$\alpha = 0.3$	723.6	1429.4	4722.3	31.8	141.5	1800.5	157.8	924.0	**4153.3**
$\alpha = 0.4$	696.6	1420.4	**4716.0**	29.8	134.5	**1776.0**	145.2	892.6	4217.7
$\alpha = 0.5$	687.4	**1402.0**	4742.1	28.0	128.8	1795.4	140.4	884.8	4319.0
$\alpha = 0.6$	**595.0**	1419.9	4845.7	25.9	122.9	1849.4	132.0	**883.2**	4492.7
$\alpha = 0.7$	677.1	1498.0	5083.6	24.2	119.7	1960.3	127.0	887.5	4831.2
$\alpha = 0.8$	700.0	1607.1	5585.9	22.1	**114.4**	2172.7	**119.0**	947.5	5458.7
$\alpha = 0.9$	776.0	1935.7	6859.3	19.8	114.7	2694.9	122.6	1134.4	6776.5
$\alpha = 1.0$	5006.6	7799.6	17268.0	**13.8**	168.3	8285.1	186.8	5147.8	16163.0

these underestimations with respect to the parameter α used to split the partitions. Assuming a number of variables $n = 25$, instances for $p = 4$, $p = 7$ and $p = 10$ are considered in both the "general" and the "nonnegative" cases. Values α from 0 to 1 are tested. The average times and average iterations are given as results of this second computational test and are summarized in the following tables. Six groups of instances (depending on p, and "general"/"nonnegative" cases) and 100 instances for each group are considered, with a grand total of 19800 problems solved. Numbers in bold emphasize the best results (lower values) in terms of iterations or computational time. Notice that, at the best of our knowledge, no detailed studies have been published regarding to the impact of the splitting parameter α in the behavior of the branch-and-bound method (for example, just $\alpha = 0.25$ is used in Gerard et al. (2017) and just $\alpha = 0.8$ is considered in Fampa et al. (2017)). Taking into account the results in Tables 3, 4, 5 and 6, it is worth noticing that:

- underestimation functions u_4 and u_7 have similar performances when $p < n$; in this light, notice that u_4 can be easily obtained while u_7 needs some eigenvectors to be computed; take into account also that in the case $p > n$ underestimation u_7 has less branching variables than u_4 and hence is more performing;
- u_0 has the best performances, but needs some eigenvectors to be computed;
- the use of $\alpha = 1.0$ should be avoided since provides bad performances;
- the greater is the value of p, the smaller is the value of α providing the best performance; in this light, the parameters suggested in Fampa et al. (2017), Gerard et al. (2017) seem no useful;
- performances in the "nonnegative" case are much better than the ones in the "general" case (underestimations' tightness results better in the "nonnegative" case than in the "general" one);

Table 8 Average elapsed times (secs)—"general" case—$p = 10$

	u_4			u_0			u_7		
	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$
$\alpha = 0.0$	7.921	15.244	55.109	0.306	1.379	21.819	1.429	9.686	48.158
$\alpha = 0.1$	7.012	14.427	53.098	0.284	1.312	21.200	1.360	9.314	47.639
$\alpha = 0.2$	6.506	13.763	52.290	0.267	1.234	20.556	1.276	8.796	46.289
$\alpha = 0.3$	6.216	13.185	50.897	0.242	1.175	19.921	1.212	8.614	**45.299**
$\alpha = 0.4$	5.779	12.906	50.196	0.226	1.105	19.336	1.093	8.187	45.416
$\alpha = 0.5$	5.552	12.487	**49.731**	0.211	1.045	**19.185**	1.033	8.012	45.796
$\alpha = 0.6$	**4.654**	**12.430**	50.110	0.197	0.989	19.419	0.956	7.814	46.850
$\alpha = 0.7$	5.148	12.780	51.611	0.182	0.950	20.220	0.908	**7.698**	49.561
$\alpha = 0.8$	5.150	13.460	55.645	0.168	0.894	22.051	**0.842**	8.047	55.044
$\alpha = 0.9$	5.539	15.880	67.482	0.150	**0.884**	26.864	0.843	9.329	67.126
$\alpha = 1.0$	31.390	60.871	167.870	**0.103**	1.222	80.116	1.161	39.250	154.790

Table 9 Average number of iterations—"nonnegative" case—$p = 10$

	u_4			u_0			u_7		
	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$
$\alpha = 0.0$	149.0	170.5	190.6	29.0	66.9	156.4	67.7	153.7	177.8
$\alpha = 0.1$	136.5	160.3	179.6	26.9	63.1	148.1	63.3	143.3	166.6
$\alpha = 0.2$	128.5	151.2	168.8	25.3	58.8	138.3	59.3	133.8	156.4
$\alpha = 0.3$	118.3	144.3	158.9	23.7	54.8	131.4	54.9	124.6	145.9
$\alpha = 0.4$	110.7	136.9	153.1	22.0	51.1	124.2	51.4	116.8	139.5
$\alpha = 0.5$	102.4	127.1	147.8	20.2	47.8	116.6	47.7	109.3	131.1
$\alpha = 0.6$	95.4	120.3	141.5	18.6	43.8	109.8	43.6	100.8	124.6
$\alpha = 0.7$	87.1	113.4	**140.4**	17.0	40.2	105.9	39.7	93.2	118.2
$\alpha = 0.8$	79.7	**109.8**	147.9	14.9	36.2	**104.6**	35.6	85.2	**116.9**
$\alpha = 0.9$	**73.9**	117.6	198.7	12.6	32.2	117.4	30.4	**79.4**	135.8
$\alpha = 1.0$	841.3	1177.7	1327.5	**4.0**	**31.9**	1201.2	**30.2**	1083.7	1346.6

- performances decrease exponentially with respect to the number of branching variables (and, hence, with respect to p);
- as regards the particular class of problems described in Sect. 4, in which u_0, u_4 and u_7 coincide, the best choice is to use u_4 which needs no eigendecompositions.

4.3 A deep comparison of u_0, u_4, u_7—part 2

The aim of this subsection is to focus on the behavior of underestimations u_0, u_4 and u_7 with respect to the number of variables n. Assuming a parameter $p = 10$, instances for $n = 5$, $n = 10$ and $n = 20$ are considered in both the "general" and

Table 10 Average elapsed times (secs)—"nonnegative" case—$p = 10$

	u_4			u_0			u_7		
	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$	$p = 5$	$p = 10$	$p = 20$
$\alpha = 0.0$	1.520	1.744	2.116	0.231	0.579	1.719	0.607	1.601	2.021
$\alpha = 0.1$	1.369	1.624	1.968	0.213	0.546	1.622	0.567	1.483	1.892
$\alpha = 0.2$	1.287	1.525	1.843	0.201	0.509	1.513	0.529	1.381	1.762
$\alpha = 0.3$	1.182	1.435	1.733	0.191	0.476	1.436	0.490	1.278	1.653
$\alpha = 0.4$	1.107	1.353	1.663	0.778	0.446	1.360	0.462	1.197	1.576
$\alpha = 0.5$	1.008	1.254	1.604	0.165	0.419	1.280	0.424	1.112	1.482
$\alpha = 0.6$	0.928	1.176	1.528	0.151	0.389	1.208	0.387	1.014	1.406
$\alpha = 0.7$	0.843	1.097	**1.511**	0.140	0.359	1.158	0.355	0.940	1.330
$\alpha = 0.8$	0.767	**1.055**	1.577	0.125	0.329	**1.151**	0.317	0.855	**1.313**
$\alpha = 0.9$	**0.692**	1.102	2.095	0.109	0.295	1.276	0.272	**0.783**	1.506
$\alpha = 1.0$	6.271	9.913	13.334	**0.046**	**0.285**	12.185	**0.254**	9.104	13.549

the "nonnegative" cases. Values α from 0 to 1 are tested. The average times and average iterations are given as results of this second computational test and are summarized in the following tables. Six groups of instances (depending on n, and "general"/"nonnegative" cases) and 100 instances for each group are considered, with a grand total of 19800 problems solved. Numbers in bold emphasize the best results (lower values) in terms of iterations or computational time. In other words, a detailed computational experience is provided to show the behavior of underestimations u_0, u_4 and u_7 in the cases $n < p$, $n = p$ and $n > p$. The results provided in Tables 7, 8, 9 and 10 point out that:

- Underestimation u_0 is the most performing one (but needs eigenvalues and eigenvectors to be computed);
- Comparing u_4 and u_7 (recall that u_7 is derived from u_4 by means of an eigendecomposition), performances are similar when $n > p$, u_7 is better than u_4 when $n = p$, while u_7 outperforms u_4 when $n < p$; this behavior yields from the number of splitting variables which results to be smaller than or equal to $\min\{n, p\}$;
- Performances in the "nonnegative" case are much better than the ones in the "general" case;
- The higher is the value of n the smaller the value of α should be.

4.4 Overall comments

The main results of this computational experience are:

- u_4 is the most performing underestimation function based on the structure of linear multiplicative functions thus avoiding the numerical troubles of eigenvectors computing;

- Linear underestimations, often used in the literature, are actually worse than quadratic underestimations;
- The parameter α has no value better than others, unlike the literature proposes;
- In the particular applicative case described in Sect. 4 u_4 is the best choice and there is no need at all to use eigenvectors;
- The "nonnegative" case results to have better performances than the "general" one, and this deserves to be deepened on in future researches.

5 Toward applications to bilevel problems

In Sect. 3.4, a special linear multiplicative problem is considered in order to point out its behavior with respect to the underestimation functions previously introduced. At the same time, the study of this particular case is also motivated by the fact that the structure (6) is commonly present in several bilevel programming problems of *leader-follower type* (see, e.g., Dempe 2020 for a state of the art). Specifically speaking, leader-follower type problems are hierarchical mathematical formulations with an upper and lower-level structure: the upper/leader-level is a suitable optimization problem partially constrained by a second (parametric) optimization problem as lower/follower-level.

In many real-world applications formulated as bilevel programming problems, the upper/leader-level objective function is defined as in (6), that is, $f(z,\beta,g)=\sum_{i=1}^{p}\beta_i g_i+q(z,\beta,g)$. By using the notation introduced in Sect. 3.4, if $\beta:=(\beta_i)_{i=1,\dots,p}\in\mathbb{R}^p$ represents the upper/leader variable (for instance, a vector of clearing prices), $g:=(g_i)_{i=1,\dots,p}\in\mathbb{R}^p$ represents the lower/follower variable (for instance, a vector of certain quantities sold) and $x:=(z,\beta)$ so that $f(z,\beta,g)\equiv f(x,g)$, then the standard form of the resulting *optmistic bilevel programming problem* is the following:

$$\begin{aligned}&\min_{x,g} f(x,g)\\ &\text{s.t.}\ \begin{cases} x\in X\\ g\in\left\{g\in K(x)\,:\,h(x,g)=\min_{\tilde{g}\in K(x)}h(x,\tilde{g})\right\},\end{cases}\end{aligned}\tag{7}$$

where

$$X:=\left\{x\in\mathbb{R}^{p+m}\,:\,\gamma_{i_u}(x)\le 0\,,\ i_u=1,\dots,k_u\right\}$$

and

$$K(x):=\left\{g\in\mathbb{R}^{p}\,:\,\zeta_{i_l}(x,g)\le 0\,,\ i_l=1,\dots,k_l\right\}\quad\forall x\in X$$

are the upper/leader and the lower/follower feasible sets, respectively. Under suitable assumptions (see, e.g., Dempe and Zemkoho 2012, 2013) and under opportune constraints qualification conditions (see, e.g., Aussel and Svensson 2019; Dempe 2020), the optimistic bilevel programming problem (7) can be transformed into a

single-level optimization problem[1], that is, a Mathematical Program with Equilibrium Constraints (MPEC) in which the lower/follower problem is reformulated by using opportune Karush-Kuhn-Tucker (KKT) optimality conditions. Notice that, in this case, the lower-level KKT conditions result to be:

$$\begin{cases} \nabla_g h(x,g) + \nabla_g \zeta(x,g)^T \eta = 0, \\ \eta \geqq 0, \quad -\zeta(x,g) \geqq 0, \quad \eta^T \zeta(x,g) = 0. \end{cases} \tag{8}$$

Then, the resulting *KKT reformulation* of the optimistic bilevel programming problem (7) is:

$$\begin{aligned} &\min_{x,g,\eta} f(x,g) \\ &\text{s.t.} \begin{cases} x \in X \\ (g,\eta) \in KKT(x) := \{(g,\eta) : \text{ conditions (8) are satisfied }\} \end{cases} \end{aligned} \tag{9}$$

Remark 7 The KKT reformulation allows to transform the optimistic bilevel programming problem (7) into a single-level program (9) at the cost of introducing the new variables η in the formulation of the problem. In addition, the presence of complementarity constraints implies that the feasible region of the resulting problem is no more a polyhedron.

From a computational point of view, the linear multiplicative function f can be studied by using opportunely the results obtained in Sect. 3.

Moreover, the complementarity constraints of the lower/follower problem in the KKT reformulation merged with the upper/leader-level, can be managed in various ways:

- If the upper and lower level of (7) are linear multiplicative, then a classical procedure to solve bilevel programs is to consider a suitable penalization of $\eta^T \zeta(x,g)$ that could be used to get a *linear multiplicative single-level optimization problem* (see Section 2.4.3 in Bard 1997, for example) at the cost of introducing the new penalty parameter in the formulation of the problem;
- An outer approximation branch-and-bound method based on a feasible region relaxation where some of the products $\eta_{i_l} \zeta_{i_l}(x,g) = 0$ are omitted and some of the variables η_{i_l} or $\zeta_{i_l}(x,g)$ are fixed to zero;
- A quadratic indefinite penalty function $M \sum_{i_l=1}^{k_l} \eta_{i_l} \zeta_{i_l}(x,g)$, with $M >> 0$ great enough, to be added to the objective function (again, a branch-and-bound may deserve);
- Binary $0-1$ variables and the so called big-M method (constraints $\eta_{i_l} \zeta_{i_l}(x,g) = 0$ substituted with $\eta_{i_l} \leq \delta_{i_l} M$ and $\zeta_{i_l}(x,g) \leq (1-\delta_{i_l})M$, $\delta_{i_l} \in \{0,1\}$ and $M >> 0$

[1] Classical requirements on the (parametric) lower/follower-level problem are the differentiability and convexity of $h(x,\cdot)$ and the differentiability and linearity/convexity/generalized convexity of $\zeta_{i_l}(x,\cdot)$, for all $i = 1,\ldots,k_l$.

great enough), a branch-and-bound approach may be needed to manage the binary variables.

Remark 8 Complementarity constraints can be directly managed by means of a branch and bound approach where various subproblems are solved. Moreover, further approaches can be used in the case the available solvers are able to manage particular constraints. In this light, as $\eta_{i_l} \geq 0$ and $-\zeta_{i_l}(x,g) \geq 0$, then the following conditions (i)–(v) are equivalent:

(i) $\eta_{i_l}\zeta_{i_l}(x,g) = 0$;
(ii) $\left(\eta_{i_l} + \zeta_{i_l}(x,g)\right)^2 = \left(\eta_{i_l} - \zeta_{i_l}(x,g)\right)^2$;
(iii) $\left(\eta_{i_l} + \zeta_{i_l}(x,g)\right) = \left|\eta_{i_l} - \zeta_{i_l}(x,g)\right|$;
(iv) $\left(\eta_{i_l} + \zeta_{i_l}(x,g)\right) = \max\left\{\eta_{i_l} - \zeta_{i_l}(x,g);0\right\} + \max\left\{\zeta_{i_l}(x,g) - \eta_{i_l};0\right\}$;
(v) $\left\{\eta_{i_l}, \zeta_{i_l}(x,g)\right\}$ is a "SOS1" set of variables (Special Ordered Set of type 1, set of variables where at most one variable may be nonzero, all others being at 0).

As a consequence, solvers able to manage quadratic indefinite constraints, absolute value constraints, "max" constraints or SOS constraints could be useful too.

In the context of this class of problems, although opportune numerical experiments could be of great interest to provide a performance comparison with works on the topic available in the literature (see Kleinert and Schmidt 2021, for example), it is beyond the scope of this work. Anyway, future investigations will be in this direction in relation to the study of suitable real-world problems.

6 Conclusions

In this paper, an extended computational experience regarding linear multiplicative problems is provided and a unifying framework to approach such problems with a branch-and-bound method is fully described. In this light, several underestimation functions are studied and various partitioning criteria are compared. In particular, it has been shown that the quadratic underestimation function u_4 has to be chosen to avoid eigenvectors-based approaches. As regards the splitting parameter α, as higher the number of branching variables (or the number of variables) as smaller the value of α should be. The proposed solution method results to be very efficient in the case of nonnegative variables and nonnegative branching variables. In this light, a particular case (useful in applications and in bilevel programming) has been theoretically studied in deep, pointing out also that u_4 is the underestimation to be used. The results obtained within the proposed unifying framework provide detailed comparisons and improvements with respect to the current literature.

Acknowledgements Careful reviews by the anonymous referees are gratefully acknowledged.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement.

References

Aussel D, Svensson A (2019) Towards tractable constraint qualifications for parametric optimisation problems and applications to generalised nash games. J Optim Theory Appl 182:404–416. https://doi.org/10.1007/s10957-019-01529-4

Bajaj I, Faruque Hasan MM (2020) Global dynamic optimization using edge-concave underestimator. J Glob Optim 77:487–512. https://doi.org/10.1007/s10898-020-00883-2

Bard JF (1997) Practical bilevel optimization: algorithms and applications. Kluwer Academic Publishers, Alphen aan den Rijn

Cambini A, Martein L (2009) Generalized convexity and optimization: theory and applications. Lecture notes in economics and mathematical systems. Springer, Berlin

Cambini R, Sodini C (2005) Decomposition methods for solving nonconvex quadratic programs via branch and bound. J Glob Optim 33:313–336. https://doi.org/10.1007/s10898-004-6095-8

Cambini R, Sodini C (2008) A computational comparison of some branch and bound methods for indefinite quadratic programs. Cent Eur J Oper 16:139–152. https://doi.org/10.1007/s10100-007-0049-4

Cambini R, Salvi F (2009) A branch and reduce approach for solving a class of low rank d.c. programs. J Comput Appl Math. 233:492–501. https://doi.org/10.1016/j.cam.2009.07.053

Cambini R, Salvi F (2010) Solving a class of low rank d.c. programs via a branch and bound approach: a computational experience. Oper Res Lett. 38:354–357. https://doi.org/10.1016/j.orl.2010.07.008

Dempe S (2020) Bilevel optimization: theory, algorithms, applications and a bibliography. Springer, Berlin

Dempe S, Zemkoho AB (2012) On the Karush–Kuhn–Tucker reformulation of the bilevel optimization problem. Nonlinear Anal Theory Methods Appl 75:1202–1218. https://doi.org/10.1016/j.na.2011.05.097

Dempe S, Zemkoho AB (2013) The bilevel programming problem: reformulations, constraint qualifications and optimality conditions. Math Program Ser A 138:447–473. https://doi.org/10.1007/s10107-011-0508-5

Fampa M, Lee J, Melo W (2017) On global optimization with indefinite quadratics. EURO J Comput Optim 5:309–337. https://doi.org/10.1007/s13675-016-0079-6

Gerard D, Köppe M, Louveaux Q (2017) Guided dive for the spatial branch-and-bound. J Glob Optim 68:685–711. https://doi.org/10.1007/s10898-017-0503-3

Gupta OK (1995) Applications of quadratic programming. J Inf Optim Sci 16:177–194. https://doi.org/10.1080/02522667.1995.10699213

Horst R, Pardalos PM (1995) Handbook of global optimization, nonconvex optimization and its applications. Kluwer Academic Publishers, Dordrecht

Horst R, Tuy H (1996) Global optimization: deterministic approaches, 3rd edn. Springer, Berlin

Horst R, Pardalos PM, Thoai NV (2001) Introduction to global optimization, nonconvex optimization and its applications, 2nd edn. Kluwer Academic Publishers, Dordrecht

Jiao H, Liu S, Chen Y (2012) Global optimization algorithm for a generalized linear multiplicative programming. J Appl Math Comput 40:551–568

Jiao H, Wang W, Shang Y (2023) Outer space branch-reduction-bound algorithm for solving generalized affine multiplicative problems. J Comput Appl Math 419:114784

Kleinert T, Schmidt M (2021) Computing feasible points of bilevel problems with a penalty alternating direction method. INFORMS J Comput 33:198–215. https://doi.org/10.1287/ijoc.2019.0945
Konno H, Kuno T (1992) Linear multiplicative programming. Math Program 56:51–64. https://doi.org/10.1007/BF01580893
McCarl BA, Moskowitz H, Furtan H (1977) Quadratic programming applications. Omega 5:43–55. https://doi.org/10.1016/0305-0483(77)90020-2
McCormick GP (1976) Computability of global solutions to factorable nonconvex solutions: Part I: convex underestimating problems. Math Program 10:147–175. https://doi.org/10.1007/BF01580665
Mjelde KM (1983) Methods of the allocation of limited resources. Wiley, New York
Ryoo HS, Sahinidis NV (2003) Global optimization of multiplicative programs. J Glob Optim 26:387–418. https://doi.org/10.1023/A:1024700901538
Shen P, Wang K, Lu T (2020) Outer space branch and bound algorithm for solving linear multiplicative programming problems. J Glob Optim 78:453–482. https://doi.org/10.1007/s10898-020-00919-7
Shen P, Wang K, Lu T (2022) Global optimization algorithm for solving linear multiplicative programming problems. Optimization 71:1421–1441. https://doi.org/10.1080/02331934.2020.1812603
Tuy H (2016) Convex analysis and global optimization, 2nd edn. Springer optimization and its applications. Springer, Berlin
Wang CF, Liu SY, Shen P (2012) Global minimization of a generalized linear multiplicative programming. Appl Math Model 36:2446–2451. https://doi.org/10.1016/j.apm.2011.09.002
Zhou XG, Cao BY, Wu K (2015) Global optimization method for linear multiplicative programming. Acta Math Sin 31:325–334. https://doi.org/10.1007/s10255-015-0456-6