Hindawi
Complexity
Volume 2024, Article ID 9753818, 1 page
https://doi.org/10.1155/2024/9753818

Retraction

Retracted: Branch-and-Reduction Algorithm for Indefinite Quadratic Programming Problem

Complexity

Received 23 January 2024; Accepted 23 January 2024; Published 24 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Y. Qiu, Y. Zhu, and J. Yin, "Branch-and-Reduction Algorithm for Indefinite Quadratic Programming Problem," *Complexity*, vol. 2021, Article ID 5578427, 14 pages, 2021.

Hindawi
Complexity
Volume 2021, Article ID 5578427, 14 pages
https://doi.org/10.1155/2021/5578427

Research Article

Branch-and-Reduction Algorithm for Indefinite Quadratic Programming Problem

Yongjian Qiu, [1,2] Yuming Zhu, [1] and Jingben Yin [3]

[1] *School of Management, Northwestern Polytechnical University, Xi'an 710072, China*
[2] *School of Clinical, Xinxiang Medical University, Xinxiang 453003, China*
[3] *School of Mathematical Sciences, Henan Institute of Science and Technology, Xinxiang 453003, China*

Correspondence should be addressed to Yongjian Qiu; 05091002@sqmc.edu.cn

Received 2 February 2021; Revised 3 March 2021; Accepted 18 March 2021; Published 30 March 2021

Academic Editor: Zhihan Lv

This paper presents a rectangular branch-and-reduction algorithm for globally solving indefinite quadratic programming problem (IQPP), which has a wide application in engineering design and optimization. In this algorithm, first of all, we convert the IQPP into an equivalent bilinear optimization problem (EBOP). Next, a novel linearizing technique is presented for deriving the linear relaxation programs problem (LRPP) of the EBOP, which can be used to obtain the lower bound of the global optimal value to the EBOP. To obtain a global optimal solution of the EBOP, the main computational task of the proposed algorithm involves the solutions of a sequence of LRPP. Moreover, the global convergent property of the algorithm is proved, and numerical experiments demonstrate the higher computational performance of the algorithm.

1. Introduction

This paper considers the following indefinite quadratic programming problem:

$$\text{(IQPP)}: \begin{cases} \min & f(s) = s^T \Theta s + c^T s, \\ \text{s.t.} & s \in D = \{s \in R^n | As \leq b\}, \end{cases} \quad (1)$$

where $\Theta = (\Theta_{jk})_{n\times n}$ is an $n \times n$ real symmetric matrix; $A = (a_{jk})_{m\times n}$ is an $m \times n$ real matrix, $c \in R^n$, $s \in R^n$, and $b \in R^m$.

The IQPP is a class of important nonlinear and nonconvex optimization problems; it has attracted the attention of many scholars for many years. On the one hand, it is because the IQPP has a wide range of applications in management science, optimal control, financial optimization, engineering design, production plan, and so on [1, 2]. On the other hand, it is because many nonlinear and nonconvex optimization problems can be transformed into this form of the IQPP [3–6], such as linear multiplicative programs problem, generalized linear multiplicative programs problem, and 0-1 programs problem. In addition, since the IQPP usually produces many local optimum solutions which are not global optimum, which puts forward many important theories and computational difficulties, that it is very necessary to propose a feasible and effective algorithm for globally solving the IQPP.

In the last decades, many algorithms have been proposed for globally solving the IQPP and its special forms. In these algorithms, most of them adopt the branch-and-bound framework, for example, the branch-and-bound algorithm based on parametric linear relaxation [7], branch-and-bound outer approximation algorithm [8], branch-and-reduce algorithms based on linear relaxation [9–18], and branch-and-cut algorithm [19]. In these algorithms, to be specified, Gao et al. [10, 11] propose two novel branch-and-reduce approaches for the IQPP by employing a new rectangle partitioning method and some reducing tactics. By using linear relaxation approach or parametric linear relaxation method to derive the more reliable lower bound, Jiao et al. [14, 16] present two range division and reduction algorithms for the quadratically constrained sum of quadratic ratios problem, which acquire the efficient lower bounds without introducing additional new variables and constraints, which can be also used to globally solve the IQPP; by using a new parametric linearizing technique, Jiao et al. [9] give a parametric linear relaxation algorithm for globally solving nonconvex quadratic programming problem with linear constraints; by employing a new accelerating technique, Ge and Liu [12] propose an accelerating algorithm for globally solving the indefinite quadratic programming problem with quadratic constraints. In addition to the algorithms mentioned above, some algorithms for linear multiplicative programming, nonlinear multiplicative programming, generalized linear multiplicative programming, and generalized geometric programming can be also used to solve these classes of IQPP; see [4, 20–40] for details.

Although many solution algorithms have been proposed for the IQPP, the global optimization algorithm for solving the IQPP with the assumption that $\Theta = (\Theta_{jk})_{n\times n}$ is an arbitrary $n \times n$ real symmetric matrix is rarely studied in the literature. Thus, it is still very necessary to propose a potential practical algorithm for globally solving the IQPP.

In this paper, based on the branch-and-bound framework and new linear relaxation technique, we will present a new branch-and-reduction algorithm for globally solving the IQPP. Firstly, we convert the IQPP into an EBOP. Secondly, we construct a new linear relaxation technique, which can be used to derive the LRPP of the EBOP. Thirdly, a new deleting technique is constructed for reducing the scope of the investigated rectangle. Finally, the global convergence of the proposed algorithm is proved, and comparing with the known methods, numerical experimental results show that the proposed algorithm in this paper works as well as or better than the currently known methods.

The remaining sections of this paper are organized as follows: in Section 2, first of all, we convert the IQPP into the EBOP. Next, a new linearizing technique is proposed for deriving the LRPP of the EBOP. In Section 3, a reduction technique is derived for improving the computational efficiency, based on the branch-and-bound framework, by combing the derived LRPP with the deleting technique, a branch-and-reduction algorithm is established for globally solving the IQPP in Section 4, and its global convergence is derived in this section. In Section 5, comparing with the existing methods, some numerical examples in the recent literature are used to verify the feasibility and computational efficiency of the algorithm. Finally, some conclusions are presented.

2. Linear Relaxation Programming Problems

Let Θ_i be the ith row of the matrix Θ, and let

$$\begin{aligned} t_i &= \Theta_i s = \sum_{k=1}^{n} \Theta_{ik} s_k, \quad i = 1,2,\ldots,n,\\ \underline{s}_i^0 &= \min_{s\in D} s,\\ \overline{s}_i^0 &= \max_{s\in D} s, \quad i = 1,2,\ldots,n,\\ \underline{t}_i^0 &= \min_{s\in D\cap H^0} \Theta_i s,\\ \overline{t}_i^0 &= \max_{s\in D\cap H^0} \Theta_i s, \quad i = 1,2,\ldots,n,\\ H^0 &= \left\{ s \in R^n \mid \underline{s}_i^0 \le s_i \le \overline{s}_i^0, \quad i = 1,2,\ldots,n \right\},\\ T^0 &= \left\{ t \in R^n \mid \underline{t}_i^0 \le t_i \le \overline{t}_i^0, \quad i = 1,2,\ldots,n \right\}. \end{aligned} \tag{2}$$

By introducing new variables t_i, $i = 1,\ldots,n$, we can convert the IQPP into the following EBOP:

$$(\text{EBOP}): \begin{cases} \min & f(s,t) = \sum_{i=1}^{n} c_i s_i + \sum_{i=1}^{n} s_i t_i \\ & \text{As} \le b, \\ \text{s.t.} & \\ & t_i = \sum_{k=1}^{n} \theta_{ik} s_k,\ i = 1,2,\ldots,n,\ s \in H^0,\ t \in T^0. \end{cases} \tag{3}$$

In the following, the main work is to solve the EBOP. As everyone knows, in a branch-and-bound procedure, to globally solving the EBOP, the principal operation is the computation of lower bounds for the EBOP and its subproblems. The lower bounds of the global optimal values of the EBOP and its subproblems can be obtained by solving a series of LRPP of the EBOP, which can be established by the following linear relaxation technique.

Without loss of generality, we let

$$\begin{aligned} H &= \left\{ s \in R^n \mid \underline{s}_i \le s_i \le \overline{s}_i, \quad i = 1,2,\ldots,n \right\} \subseteq H^0,\\ T &= \left\{ t \in R^n \mid \underline{t}_i \le t_i \le \overline{t}_i, \quad i = 1,2,\ldots,n \right\} \subseteq T^0. \end{aligned} \tag{4}$$

And without loss of generality, for any $s \in H$, $t \in T$ and define the following functions:

$$\begin{aligned}
g(s_i) &= s_i^2,\\
g^l(s_i) &= (\underline{s}_i + \overline{s}_i)s_i - \frac{(\underline{s}_i + \overline{s}_i)^2}{4},\\
g^u(s_i) &= (\underline{s}_i + \overline{s}_i)s_i - \underline{s}_i\overline{s}_i,\\
g(t_i) &= t_i^2,\\
g^l(t_i) &= (\underline{t}_i + \overline{t}_i)t_i - \frac{(\underline{t}_i + \overline{t}_i)^2}{4},\\
g^u(t_i) &= (\underline{t}_i + \overline{t}_i)t_i - \underline{t}_i\overline{t}_i,\\
g(s_i, t_i) &= s_i t_i,\\
g^u(s_i, t_i) &= \frac{1}{2}\left[(\underline{s}_i + \overline{s}_i)t_i + (\underline{t}_i + \overline{t}_i)s_i + \frac{(\underline{s}_i - \overline{t}_i + \overline{s}_i - \underline{t}_i)^2}{4} - \underline{s}_i\overline{s}_i - \underline{t}_i\overline{t}_i\right],\\
g^l(s_i, t_i) &= \frac{1}{2}\left[(\underline{s}_i + \overline{s}_i)t_i + (\underline{t}_i + \overline{t}_i)s_i + (\underline{s}_i - \overline{t}_i)(\overline{s}_i - \underline{t}_i) - \frac{(\underline{s}_i + \overline{s}_i)^2}{4} - \frac{(\underline{t}_i + \overline{t}_i)^2}{4}\right].
\end{aligned} \tag{5}$$

As we know, the function $g(s_i) = s_i^2$ is a convex function about s_i over $[\underline{s}_i, \overline{s}_i]$; we have that its affine concave envelope is

$$g^u(s_i) = (\underline{s}_i + \overline{s}_i)s_i - \underline{s}_i\overline{s}_i, \tag{6}$$

and its tangential approximation function is

$$g^l(s_i) = (\underline{s_i} + \overline{s}_i)s_i - \frac{(\underline{s}_i + \overline{s}_i)^2}{4}, \tag{7}$$

which is parallel to $g^u(s_i)$ and the tangential approximation point occurs at $(\underline{s}_i + \overline{s}_i/2)$.

Therefore, from the geometric property of the function $g(s_i) = s_i^2$ over $[\underline{s}_i, \overline{s}_i]$, we have

$$\begin{aligned}
g^l(s_i) &= (\underline{s_i} + \overline{s_i})s_i - \frac{(\underline{s_i} + \overline{s_i})^2}{4} \le s_i^2 \le (\underline{s}_i + \overline{s}_i)s_i - \underline{s}_i\overline{s}_i\\
&= g^u(s_i).
\end{aligned} \tag{8}$$

Similarly, we consider the function $g(t_i) = t_i^2$ over $[\underline{t}_i, \overline{t}_i]$; it can follow that

$$g^l(t_i) = (\underline{t}_i + \overline{t}_i)t_i - \frac{(\underline{t}_i + \overline{t}_i)^2}{4} \le g(t_i) \le (\underline{t}_i + \overline{t}_i)t_i - \underline{t}_i\overline{t}_i = g^u(t_i). \tag{9}$$

Furthermore, if we suppose that $(s_i - t_i)$ is a single variable, then $(s_i - t_i)^2$ is a convex function about $(s_i - t_i)$ over $[\underline{s}_i - \overline{t}_i, \overline{s}_i - \underline{t}_i]$. By the above conclusions, we can get that

$$(\underline{s}_i - \overline{t}_i + \overline{s}_i - \underline{t}_i)(s_i - t_i) - \frac{(\underline{s}_i - \overline{t}_i + \overline{s}_i - \underline{t}_i)^2}{4} \le (s_i - t_i)^2, \tag{10}$$

$$(s_i - t_i)^2 \le (\underline{s}_i - \overline{t}_i + \overline{s}_i - \underline{t}_i)(s_i - t_i) - (\underline{s}_i - \overline{t}_i)(\overline{s}_i - \underline{t}_i). \tag{11}$$

By (8)–(11), it follows that

$$\begin{aligned} g(s_i,t_i) &= \frac{1}{2}\left[s_i^2+t_i^2-(s_i-t_i)^2\right] \geq \frac{1}{2}\left[(\underline{s}_i+\overline{s}_i)s_i-\frac{(\underline{s}_i+\overline{s}_i)^2}{4}\right]+\frac{1}{2}\left[(\underline{t}_i+\overline{t}_i)t_i-\frac{(\underline{t}_i+\overline{t}_i)^2}{4}\right] \\ &\quad -\frac{1}{2}\left[(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)(s_i-t_i)-(\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i)\right] \\ &= \frac{1}{2}\left[(\underline{t}_i+\overline{t}_i)s_i+(\underline{s}_i+\overline{s}_i)t_i-\frac{(\underline{t}_i+\overline{t}_i)^2}{4}-\frac{(\underline{s}_i+\overline{s}_i)^2}{4}+(\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i)\right] = g^l(s_i,t_i), \\ g(s_i,t_i) &= \frac{1}{2}\left[s_i^2+t_i^2-(s_i-t_i)^2\right] \leq \frac{1}{2}\left[(\underline{s}_i+\overline{s}_i)s_i-\underline{s}_i\overline{s}_i+(\underline{t}_i+\overline{t}_i)t_i-\underline{t}_i\overline{t}_i\right] \\ &\quad -\frac{1}{2}\left[(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)(s_i-t_i)-\frac{(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)^2}{4}\right] \\ &= \frac{1}{2}\left[(\underline{s}_i+\overline{s}_i)t_i+(\underline{t}_i+\overline{t}_i)s_i+\frac{(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)^2}{4}-\underline{s}_i\overline{s}_i-\underline{t}_i\overline{t}_i\right] = g^u(s_i,t_i). \end{aligned} \tag{12}$$

Thus, we have that

$$g^l(s_i,t_i) \leq g(s_i,t_i) \leq g^u(s_i,t_i). \tag{13}$$

In the following, for any $s \in H \subseteq H^0, t \in T^0$, without loss of generality, we define the function

$$f^L(s,t) = \sum_{i=1}^{n} c_i s_i + \sum_{i=1}^{n} g^l(s_i,t_i). \tag{14}$$

By the above conclusions, for any $s \in H \subseteq H^0, t \in T^0$, it is obvious that we have

$$f^L(s,t) \leq f(s,t). \tag{15}$$

Thus, based on the former discussions, we can construct the corresponding LRPP of the EBOP as follows:

$$(\text{LRPP}): \begin{cases} \min & f^L(s,t) = \sum_{i=1}^{n} c_i s_i + \sum_{i=1}^{n} g^l(s_i,t_i), \\ & As \leq b, \\ \text{s.t.} & t_i = \sum_{k=1}^{n} \Theta_{ik} s_k,\ i = 1,2,\ldots,n,\ s \in H,\ t \in T^0. \end{cases} \tag{16}$$

Remark 1. Based on the above constructing method of the LRPP, it is obvious that each feasible solution of the IQPP over subrectangle H is also feasible to the LRPP, and the global optimal value of the LRPP is less than or equal to that of the EBOP over subrectangle H. Thus, the LRPP can provide a reliable lower bound for the global optimal value of the IQPP over subrectangle H.

3. New Rectangular Reduction Technique

For improving the convergent speed of the algorithm, in this section, we will construct a new rectangular reduction technique. Without loss of generality, for any rectangle $H \subseteq H^0$, we want to investigate whether or not H contains the global optimal solution of the EBOP over H^0. The detailed discussions are given by Theorem 1.

Theorem 1. *Suppose that UB is a known upper bound of the global optimal value of the EBOP, for any subrectangle $H \subseteq H^0$; then, we have the following conclusions:*

(a) *If $\sum_{i=1}^{n} \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^{n} \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} >$ UB, then there exists no global optimal solution of the EBOP.*

(b) *If $\sum_{i=1}^{n} \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^{n} \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} \leq$ UB, then we get, for each $\tau \in \{1,2,\ldots,n\}$, if $c_\tau > 0$, then the subrectangle $\overline{H} = \{s \in R^n | \overline{s}_i \leq s_i \leq \overline{s}_i,\ i \neq \tau; \rho_\tau < s_\tau \leq \overline{s}_\tau\}$ does not include any global optimal* *solution of the EBOP; if $c_\tau < 0$, then the subrectangle $\overline{\overline{H}} = \{s \in R^n | \underline{s}_i \leq s_i \leq \overline{s}_i,\ i \neq \tau; \underline{s}_\tau \leq s_\tau < \rho_\tau\}$ does not include any global optimal solution of the EBOP, where*

$$\begin{aligned}
\rho_\tau &= \frac{\mathrm{UB} - \left(\sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\}\right) + \min\{c_\tau\underline{s}_\tau, c_\tau\overline{s}_\tau\}}{c_\tau}, \quad \tau = 1, \ldots, n, \\
\underline{t}_i &= \min_{s \in D \cap H} \Theta_i s, \\
\overline{t}_i &= \max_{s \in D \cap H} \Theta_i s, \quad i = 1, 2, \ldots, n.
\end{aligned} \tag{17}$$

Proof

(a) For any $s \in H$, if $\sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} > \mathrm{UB}$, then we have

$$f(s,t) = \sum_{i=1}^n c_i s_i + \sum_{i=1}^n s_i t_i \geq \sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} > \mathrm{UB}. \tag{18}$$

Therefore, the subrectangle H does not produce the global optimal solution for the EBOP.

(b) If $\sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} \leq \mathrm{UB}$, for each $\tau \in \{1, 2, \ldots, n\}$, then we have the following conclusions:

(i) If $c_\tau > 0$, for any $s \in \overline{H}$, we have $s_\tau > \rho_\tau$, that is, $c_\tau s_\tau > \mathrm{UB} - \left(\sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\}\right) + \min\{c_\tau\underline{s}_\tau, c_\tau\overline{s}_\tau\}$. Moreover, we can follow that

$$\begin{aligned}
f(s,t) &= \sum_{i=1, i\neq\tau}^n c_i s_i + c_\tau s_\tau + \sum_{i=1}^n s_i t_i \geq \sum_{i=1, i\neq\tau}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + c_\tau s_\tau + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} \\
&> \sum_{i=1, i\neq\tau}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\} + \mathrm{UB} \\
&\quad - \left(\sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\}\right) + \min\{c_\tau\underline{s}_\tau, c_\tau\overline{s}_\tau\} = \mathrm{UB}.
\end{aligned} \tag{19}$$

That is, we have $f(s,t) > \mathrm{UB}$. Therefore, the subrectangle $\overline{H}$ does not include the global optimal solution of the EBOP.

(ii) : If $c_\tau < 0$, then, for any $s \in \overline{\overline{H}}$, we have $s_\tau < \rho_\tau$, that is, $c_\tau s_\tau > \mathrm{UB} - \left(\sum_{i=1}^n \min\{c_i\underline{s}_i, c_i\overline{s}_i\} + \sum_{i=1}^n \min\{\underline{s}_i\underline{t}_i, \underline{s}_i\overline{t}_i, \overline{s}_i\underline{t}_i, \overline{s}_i\overline{t}_i\}\right) + \min\{c_\tau\underline{s}_\tau, c_\tau\overline{s}_\tau\}$. Similar to the proof of the above case (i), it follows that $f(s,t) > \mathrm{UB}$; thus, the subrectangle $\overline{\overline{H}}$ does not include the global optimal solution of the EBOP, and the proof is completed. □

From Theorem 1, we can construct a rectangular reduction technique to delete a part of the investigated subrectangle H, which does not include the global optimal solution of the EBOP.

4. New Branch-and-Reduction Algorithm

In this section, combining the former LRPP and the rectangular reduction technique, we will construct a new branch-and-reduction algorithm to globally solve the IQPP.

4.1. Branching Method. In this paper, we will adopt a standard rectangular bisection method, which is adequate to guarantee the global convergence of the proposed algorithm; the detailed branching idea is given as follows.

Consider the selected subrectangle $H' = [\underline{s}', \overline{s}'] \subseteq H^0$, and denote by $q \in \arg\max\{\overline{s}'_i - \underline{s}'_i : i = 1, 2, \ldots, n\}$; then, we can partition H' into two subrectangles H'_1 and H'_2 by subdividing $[\underline{s}'_q, \overline{s}'_q]$ into $[\underline{s}'_q, (\underline{s}'_q + \overline{s}'_q/2)]$ and $[(\underline{s}'_q + \overline{s}'_q/2), \overline{s}'_q]$. By the above branching idea, we can see that the interval $[\underline{t}', \overline{t}']$ of t never is partitioned by the branching operation; it is to say, the branching process only takes place in a space of n dimension.

4.2. Branch-and-Reduction Algorithm. Without loss of generality, we assume that $f^L(s^r(H), t^r(H))$ and $(s^r(H), t^r(H))$ be the global optimal value and the global optimal solution of the problem (LRPP) over the subrectangle H, respectively. Combining the former linear relaxation method and branching method with the rectangular reduction technique together, we can establish a branch-and-reduction algorithm for globally solving the IQPP as follows:

(i) Branch-and-reduction algorithm:

(ii) Step 0 (initialization). Initialize $k := 0$, $\Lambda_0 = \{H^0\}$, $\varepsilon = 10^{-6}$, and $F := \varnothing$.

(iii) Solve the LRPP over H^0 to obtain its corresponding optimal solution $s^0 := s(H^0)$, $t^0 := t(H^0)$ and optimal value $\mathrm{LB}_0 := f^L(s^0, t^0)$. Let $\mathrm{UB} = f(s^0, t^0)$ and $F = F \cup \{(s^0, t^0)\}$. If $\mathrm{UB} - \mathrm{LB}_0 \le \varepsilon$, then the algorithm stops, and we get that (s^0, t^0) is an ε–global optimum solution for the IQPP. Otherwise, proceed with Step 1.

(iv) Step 1 (regional reduction). For each selected subrectangle H^k, use the former rectangular reduction technique of Section 3 to compress its range and still denote by the remaining subrectangle H_k.

(v) Step 2 (regional partition). Partitioning the selected rectangle H^k into two new subrectangles, denote by the set of new subdivided subrectangles $\widehat{H}^k$.

(vi) Step 3 (updating the lower bound and the upper bound). For each $H \in \widehat{H}^k$, solve the LRPP over H to obtain its corresponding optimal value $f^L(s^r(H), t^r(H))$ and optimal solution $(s^r(H), t^r(H))$. Let $\mathrm{LB}(H) := f^L(s^r(H), t^r(H))$. If $\mathrm{LB}(H) > \mathrm{UB}$, then let $\widehat{H}^k := \widehat{H}^k \setminus H$. Denote bythe residual partitioning set $\Lambda_k := (\Lambda_k \setminus H^k) \cup \widehat{H}^k$, and update the lower bound by $\mathrm{LB}_k: = \inf_{H\in\Lambda_k} \mathrm{LB}(H)$.

(vii) If the midpoint s_{mid} of H is feasible to the EBOP, then let $t_{\mathrm{mid}} = Qs_{\mathrm{mid}}$ and $F := F \cup \{(s_{\mathrm{mid}}, t_{\mathrm{mid}})\}$. Moreover, since $(s^r(H)), (t^r(H))$ is always feasible to the EBOP, we let $F := F \cup \{(s^r(H)), t^r(H)\}$.

(viii) At the same time, we update the upper bound by $\mathrm{UB}: = \min_{(s,t)\in F} f(s,t)$, and denote by the known best feasible solution $(s_{\mathrm{best}}, t_{\mathrm{best}}) := \operatorname{argmin}_{(s,t)\in F} f(s,t)$.

(ix) Step 4 (termination judgment). If $\mathrm{UB} - \mathrm{LB}_k \le \varepsilon$, the algorithm stops, and UB and s_{best} are the ε–global optimal value and the global optimal solution of the IQPP, respectively. Otherwise, let $k: = k+1$, select a new active node H^{k+1} satisfying $H^{k+1} = \operatorname{argmin}_{H\in\Lambda_k} \mathrm{LB}(H)$, and return to step 1.

4.3. Global Convergence of the Proposed Algorithm. In the section, the global convergence of the proposed algorithm is discussed as follows.

If the proposed algorithm terminates going through finite iterations, without loss of generality, we assume that it terminates at the k_{th} iteration, $k \ge 0$. By steps of the former algorithm, we can obtain that $\mathrm{UB} - \mathrm{LB}_k \le \varepsilon$. By the method of the updating upper bound, there must exist a feasible solution s^* of the IQPP such that $\mathrm{UB} = g(s^*)$. By the structure of the branch-and-reduction algorithm, we have that $\mathrm{LB}_k \le v$. Since s^* is a feasible solution for the IQPP, it follows that $f(s^*) \ge v$. Combining the above inequalities and equalities together, it follows that $v \le f(s^*) \le \mathrm{LB}_k + \varepsilon \le v + \varepsilon$. Thus, we can get that s^* is a global ε– optimum solution for the IQPP. If the proposed algorithm does not terminate after finite iterations, then we have the following conclusions.

Theorem 2 *(i) The functional differences* $\triangle(s_i, t_i) = g(\underline{s_i}, t_i) - g^l(s_i, t_i) \longrightarrow 0$ *and* $\overline{\triangle}(s_i, t_i) = g^u(s_i, t_i) - g(s_i, t_i) \longrightarrow 0$, *as* $\|\overline{s} - \underline{s}\| \longrightarrow 0$.

(ii) The functional difference $f(s,t) - f^L(s,t) \longrightarrow 0$ *as* $\|\overline{s} - \underline{s}\| \longrightarrow 0$.

(iii) If the proposed algorithm does not finitely terminate, then the proposed algorithm generates an infinite sequence $\{s^k\}$ *of which any accumulation point will be a global optimal solution of the IQPP.*

Proof (i) Without loss of generality, we define the following functional differences:

$$\begin{aligned}\triangle(s_i) &= g(s_i) - g^l(s_i),\\ \overline{\triangle}(s_i) &= g^u(s_i) - g(s_i),\\ \triangle(t_i) &= g(t_i) - g^l(t_i),\\ \overline{\triangle}(t_i) &= g^u(t_i) - g(t_i).\end{aligned} \tag{20}$$

(ii) Since $\triangle(s_i)$ is a convex function of s_i over $[\underline{s}_i, \overline{s}_i]$, it follows that $\triangle(s_i)$ can attain maximum value at the point $\underline{s}_i$ or $\overline{s}_i$. Then,

$$\begin{aligned}\max_{s_i\in[\underline{s}_i,\overline{s}_i]} \triangle(s_i) &= g(\overline{s}_i) - g^l(\overline{s}_i)\\ &= g(\underline{s}_i) - g^l(\underline{s}_i) = \frac{(\overline{s}_i - \underline{s}_i)^2}{4}.\end{aligned} \tag{21}$$

(iii) On the other hand, since $\triangle(s_i)$ is a concave function of s_i over $[\underline{s}_i, \overline{s}_i]$, $\triangle(s_i)$ can attain maximum value at the point, that is,

$$\max_{s_i\in[\underline{s}_i,\overline{s}_i]} \overline{\triangle}(s_i) = g^u\left(\frac{\underline{s}_i+\overline{s}_i}{2}\right) - g\left(\frac{\underline{s}_i+\overline{s}_i}{2}\right) = \frac{(\overline{s}_i-\underline{s}_i)^2}{4}. \tag{22}$$

(iv) So, we have

$$\max_{s_i\in[\underline{s}_i,\overline{s}_i]} \triangle(s_i) = \max_{s_i\in[\underline{s}_i,\overline{s}_i]} \overline{\triangle}(s_i) \longrightarrow 0, \text{ as } \left|\overline{s}_i-\underline{s}_i\right| \longrightarrow 0. \tag{23}$$

(v) That is to say, $\triangle(s_i), \overline{\triangle}(s_i) \longrightarrow 0$, as $\|\overline{s}-\underline{s}\| \longrightarrow 0$.

(vi) Similar to the above proof, we can also prove that

$$\max_{t_i\in[\underline{t}_i,\overline{t}_i]} \triangle(t_i) = \max_{t_i\in[\underline{t}_i,\overline{t}_i]} \overline{\triangle}(t_i) \longrightarrow 0, \text{ as } \left|\overline{t}_i-\underline{t}_i\right| \longrightarrow 0. \tag{24}$$

(vii) And $\triangle(t_i), \overline{\triangle}(t_i) \longrightarrow 0$, as $\|\overline{t}-\underline{t}\| \longrightarrow 0$. Also, since $\|\overline{t}-\underline{t}\| \longrightarrow 0$ as $\|\overline{s}-\underline{s}\| \longrightarrow 0$, therefore we have

$$\max_{t_i\in[\underline{t}_i,\overline{t}_i]} \triangle(t_i) = \max_{t_i\in[\underline{t}_i,\overline{t}_i]} \overline{\triangle}(t_i) \longrightarrow 0, \text{ as } \|\overline{s}-\underline{s}\| \longrightarrow 0. \tag{25}$$

(viii) Define

$$\begin{aligned}
\triangle(s_i-t_i) &= (s_i-t_i)^2 - \frac{1}{2}\left[(\underline{t}_i+\overline{t}_i)s_i + (\underline{s}_i+\overline{s}_i)t_i - \frac{(\underline{t}_i+\overline{t}_i)^2}{4} - \frac{(\underline{s}_i+\overline{s}_i)^2}{4} + (\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i)\right],\\
\overline{\triangle}(s_i-t_i) &= \frac{1}{2}\left[(\underline{t}_i+\overline{t}_i)s_i + (\underline{s}_i+\overline{s}_i)t_i - \underline{s}_i\overline{s}_i - \underline{t}_i\overline{t}_i + \frac{(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)^2}{4}\right] - (s_i-t_i)^2.
\end{aligned} \tag{26}$$

(ix) Using similar methods to the above proof, we can draw the following conclusions, as $\|\overline{s}-\underline{s}\| \longrightarrow 0$:

$$\max_{(s_i-t_i)\in[(\underline{s}_i-\overline{t}_i),(\overline{s}_i-\underline{t}_i)]} \triangle(s_i-t_i) = \max_{(s_i-t_i)\in[(\underline{s}_i-\overline{t}_i),(\overline{s}_i-\underline{t}_i)]} \overline{\triangle}(s_i-t_i) \longrightarrow 0. \tag{27}$$

(x) Consider

$$\begin{aligned}
\triangle(s_i,t_i) &= g(s_i,t_i) - g^l(s_i,t_i) = s_it_i - \frac{1}{2}\left[(\underline{s}_i+\overline{s}_i)t_i + (\underline{t}_i+\overline{t}_i)s_i - \frac{(\underline{s}_i+\overline{s}_i)^2}{4} - \frac{(\underline{t}_i+\overline{t}_i)^2}{4} + (\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i)\right]\\
&= \frac{1}{2}\left[s_i^2+t_i^2-(s_i-t_i)^2\right] - \frac{1}{2}\left[(\underline{s}_i+\overline{s}_i)t_i + (\underline{t}_i+\overline{t}_i)s_i - \frac{(\underline{s}_i+\overline{s}_i)^2}{4} - \frac{(\underline{t}_i+\overline{t}_i)^2}{4} + (\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i)\right]\\
&= \frac{1}{2}\left[s_i^2+t_i^2-(s_i-t_i)^2\right] - \frac{1}{2}\left[(\underline{s}_i+\overline{s}_i)s_i - \frac{(\underline{s}_i+\overline{s}_i)^2}{4} + (\underline{t}_i+\overline{t}_i)t_i - \frac{(\underline{t}_i+\overline{t}_i)^2}{4}\right]\\
&\quad + \frac{1}{2}\left[(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)(s_i-t_i) - (\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i)\right]\\
&= \frac{1}{2}\left\{s_i^2 - \left[(\underline{s}_i+\overline{s}_i)s_i - \frac{(\underline{s}_i+\overline{s}_i)^2}{4}\right]\right\} + \frac{1}{2}\left\{t_i^2 - \left[(\underline{t}_i+\overline{t}_i)t_i - \frac{(\underline{t}_i+\overline{t}_i)^2}{4}\right]\right\}\\
&\quad + \frac{1}{2}\left\{(\underline{s}_i-\overline{t}_i+\overline{s}_i-\underline{t}_i)(s_i-t_i) - (\underline{s}_i-\overline{t}_i)(\overline{s}_i-\underline{t}_i) - (s_i-t_i)^2\right\}\\
&= \frac{1}{2}\triangle(s_i) + \frac{1}{2}\triangle(t_i) + \frac{1}{2}\overline{\triangle}(s_i-t_i) \le \frac{1}{2}\max_{s_i\in[\underline{s}_i,\overline{s}_i]}\triangle(s_i) + \frac{1}{2}\max_{t_i\in[\underline{t}_i,\overline{t}_i]}\triangle(t_i) + \frac{1}{2}\max_{(s_i-t_i)\in[(\underline{s}_i-\overline{t}_i),(\overline{s}_i-\underline{t}_i)]}\overline{\triangle}(s_i-t_i).
\end{aligned} \tag{28}$$

(xi) From (23)–(27), we follow that

$$\triangle(s_i, t_i) \longrightarrow 0 \text{ as } \|\bar{s} - \underline{s}\| \longrightarrow 0. \tag{29}$$

(xii) Similarly, we can prove that $\triangle(s_i, t_i) \longrightarrow 0$ as $\|\bar{s} - \underline{s}\| \longrightarrow 0$.

(xiii) By conclusion (i), we have

$$\begin{aligned} f(s,t) - f^L(s,t) &= \sum_{k=1}^{n} c_i s_i + \sum_{k=1}^{n} s_i t_i - \left[\sum_{k=1}^{n} c_i s_i + \sum_{k=1}^{n} g^l(s_i, t_i)\right] \\ &= \sum_{k=1}^{n}\left[g(s_i,t_i) - g^l(s_i,t_i)\right] = \sum_{k=1}^{n} \triangle(s_i, t_i). \end{aligned} \tag{30}$$

(xiv) By (29), we have that $\triangle(s_i, t_i) \longrightarrow 0$, as $\|\bar{s} - \underline{s}\| \longrightarrow 0$.

(xv)erefore, we have that

$$f(s,t) - f^L(s,t) \longrightarrow 0 \text{ as } \|\bar{s} - \underline{s}\| \longrightarrow 0. \tag{31}$$

(xvi) When the algorithm is infinite, first of all, by the exhaustiveness of the bisection method, we have that

$$\lim_{k\longrightarrow\infty}\left\|\bar{s}^k - \overset{k}{\underline{s}}\right\| = 0. \tag{32}$$

Secondly, by conclusions (i) and (ii) of the theorem, since $\lim_{k\longrightarrow\infty}\|\bar{s}^k - \underline{s}^k\| \longrightarrow 0$, it follows that $\lim_{k\longrightarrow\infty}(\mathrm{UB} - \mathrm{LB}_k) = 0$; therefore, the bounding operation is consistent.

By Theorem 4.3 in [25], the sufficient condition of the global convergence of the branch-and-reduction algorithm is satisfied. Therefore, the algorithm is globally convergent to the optimal solution of the IQPP. □

4.4. Computational Complexity of the Algorithm

Definition 1. Assume that $H = [\underline{s}_1, \bar{s}_1] \times \cdots \times [\underline{s}_n, \bar{s}_n] \subset R^n$ be a compact subhyperrectangle, the diameter of the hyperrectangle $H \subset R^n$ is defined by

$$\begin{aligned} \delta(H) &= \max\left\{\|\alpha - \alpha'\|_2 \colon \alpha, \alpha' \in H\right\} \\ &= \sqrt{(\bar{s}_1 - \underline{s}_1)^2 + \cdots + (\bar{s}_n - \underline{s}_n)^2} \end{aligned} \tag{33}$$

Theorem 3. *For the proposed branch-and-reduction algorithm, for any given subhyperrectangle H, without loss of generality, we assume that there exist a fixed positive constant C and a given accuracy error ϵ and assume that the branching operation will eventually subdivide the subhyperrectangle H into $\eta = 2^n$ the smaller subhyperrectangles. Then, by subdividing the subhyperrectangle H, the number of iterations of the proposed branch-and-reduction algorithm in the worst case can be given as follows:*

$$\begin{aligned} &\sum_{i=0}^{r} 2^{n.i}, \quad \text{where } r = \lceil \log_2 \frac{C.\delta(H)}{\varepsilon} \rceil, \\ &\delta(H) = \max\left\{\delta(H^l)\colon l \in \{1, 2, \ldots, \eta\}\right\}. \end{aligned} \tag{34}$$

We call $O(n) = \sum_{t=0}^{r} 2^{n.i}$ to be the convergence rate of the branch-and-reduction algorithm by subdividing space R^n.

Proof. The proof of the theorem is similar to Theorem 5 in Liu et al. [41]; thus, it is omitted here.

Remark 2. From the above discussions, we know that $O(n)$ is a function of exponential growth, the branch-and-bound search of the proposed algorithm takes place in space R^n. Therefore, the proposed algorithm has the same computational complexity as that of [12, 21–23, 26].

5. Numerical Experiment

To validate the computational performance of the algorithm, several numerical examples are used to test the algorithm. This algorithm is implemented in MATLAB 2014a software in a notebook computer with Intel (R) Core (TM) i7-6700HQ CPU@2.6 GHz Processor, 12 GB RAM Memory, and Win10 Operational System. In the MATLAB program, all linear programming problems are solved by linprog solver, and we set the convergent error $\varepsilon = 10^{-6}$. For all numerical examples 1–12, compared with the current known algorithms in recent literature, numerical results are listed in Tables 1 and 2. In Table 1, "Iter" denotes the number of iterations.

Examples 1–11 are all deterministic examples with the quadratic objective function, which have deterministic global optimal solutions and optimal values. Moreover, examples 1-11 are all nonconvex optimization problems; they have multiple local minima that are not global optimal. Example 12 is a large-scale random example with large-size variables and constraints.

Example 1 (see [12, 20–22])

Table 1: Numerical comparisons for test examples 1–11.

Examples	Methods	Optimal value	Optimal solution	Iter	Time (s)
1	Ours	10.0000	(2.0000, 8.0000)	59	14.9847
	[12]	10.0000	(2.0000, 8.0000)	56	15.8054
	[20]	10.0000	(2.0000, 8.0000)	41	0.4856
	[21]	10.0000	(2.0000, 8.0000)	27	10.9297
	[22]	10.0000	(2.0000, 8.0000)	53	0.3278
	SCIP solver	10.0000	(2.0000, 8.0000)	1	0.3177
2	Ours	3.00000	(0.0000, 4.0000)	71	16.4324
	[12]	3.00000	(0.0000, 4.0000)	68	17.9814
	[23]	3.00000	(0.0000, 4.0000)	25	2.4936
	SCIP solver	3.00000	(0.0000, 4.0000)	1	0.0678
3	Ours	0.87016	(1.3148, 0.1396, 0.0, 0.4233)	1	0.2889
	[12]	0.87016	(1.3148, 0.1396, 0.0, 0.4233)	1	0.3894
	[22]	0.8902	(1.3148, 0.1396, 0.0, 0.4233)	1	0.5432
	SCIP solver	0.8902	(1.3148, 0.1396, 0.0, 0.4233)	11	0.2912
4	Ours	−16.22662	(0.0, 3.6403, 0.0, 2.9029, 1.9388, 0.0)	1	0.5949
	[12]	−16.22662	(0.0, 3.6403, 0.0, 2.9029, 1.9388, 0.0)	1	0.7498
	SCIP solver	−16.2266	(0.0, 3.6403, 0.0, 2.9029, 1.9388, 0.0)	1	0.0882
5	Ours	−3.00000	(3.0000, 3.0000)	32	6.2275
	[12]	−3.00000	(3.0000, 3.0000)	30	6.3380
	SCIP solver	−3.00000	(3.0000, 3.0000)	5	0.2438
6	Ours	−1.06250	(0.7500, 2.0000)	2	0.4150
	[12]	−1.06250	(0.7500, 2.0000)	2	0.5638
	SCIP solver	−1.0625	(0.7500, 2.0000)	1	0.1129
7	Ours	10	(2, 8)	18	4.5829
	[26]	10	(2, 8)	3	0.6745
	SCIP solver	10	(2, 8)	1	0.0369
8	Ours	4	(0, 0)	1	0.2276
	[26]	4	(0, 0)	2	0.4219
	SCIP solver	4	(0, 0)	1	0.1203
9	Ours	−4558	(27, 20)	35	7.4568
	SCIP solver	−4558	(27, 20)	1	0.0359
10	Ours	2.39991	(0.5828, 0.6969, 0.7515, 0.2685)	1	0.5809
	SCIP solver	2.39991	(0.5828, 0.6969, 0.7515, 0.2685)	11	0.6186
11	Ours	6.96376	(0.7767, 0.7497, 0.9079, 0.0532)	1	0.5354
	SCIP solver	6.96376	(0.7767, 0.7497, 0.9079, 0.0532)	14	0.6093

Table 2: Comparison of numerical results for example 12.

(n, m)	Ge and Liu [12]		Our algorithm		SCIP Solver	
	Avg.NT	Avg.Time	Avg.NT	Avg.Time	Avg.NT	Avg.Time
(5, 5)	1.2	0.5130	1.0	0.2592	25.9	0.7789
(10, 10)	1.2	1.1521	1.0	0.8972	606.9	13.7890
(15, 10)	1.3	4.6958	1.2	2.8975	6250.4	64.2376
(15, 15)	1.3	4.7953	1.2	2.9978	–	–
(20, 20)	1.3	8.8976	1.3	8.8840	–	–
(30, 30)	1.3	8.9189	1.3	8.9678	–	–
(40, 40)	1.2	8.4678	1.2	8.4076	–	–
(50, 50)	2.5	21.4589	2.3	19.4939	–	–
(80, 80)	2.4	33.4678	2.2	31.8257	–	–
(100, 50)	3.6	29.3849	3.5	24.9980	–	–
(100, 100)	5.4	64.8937	4.2	59.5274	–	–
(200, 20)	3.2	28.7896	3.1	26.3436	–	–
(200, 50)	3.2	42.4568	3.1	40.4673	–	–
(300, 50)	3.5	89.8947	3.3	79.7875	–	–
(300, 100)	4.8	436.3420	4.3	365.8972	–	–
(500, 50)	3.5	196.3876	3.5	193.8970	–	–
(1000, 50)	3.3	1200.7856	3.2	1154.5867	–	–

$$\begin{cases} \min & (s_1+s_2)(s_1-s_2+7) \\ & 2s_1+s_2\le 14, \\ & s_1+s_2\le 10, \\ & -4s_1+s_2\le 0, \\ & 2s_1+s_2\ge 6, \\ \text{s.t.} & s_1+2s_2\ge 6, \\ & s_1-s_2\le 3, \\ & s_1\le 5, \\ & s_1+s_2\ge 0, \\ & s_1-s_2+7\ge 0. \end{cases} \tag{35}$$

Example 2 (see [12, 23])

$$\begin{cases} \min & s_1+(2s_1-3s_2+13)(s_1+s_2-1) \\ & -s_1+2s_2\le 8, \\ & -s_2\le -3, \\ \text{s.t.} & s_1+2s_2\le 12, \\ & s_1-2s_2\le -5, \\ & s_1,s_2\ge 0. \end{cases} \tag{36}$$

Example 3 (see [12, 22])

$$\begin{cases} \min & (0.813396s_1+0.67440s_2+0.305038s_3+0.129742s_4+0.217796) \\ & \times(0.224508s_1+0.063458s_2+0.932230s_3+0.528736s_4+0.091947) \\ & 0.488509s_1+0.063565s_2+0.945686s_3+0.210704s_4\le 3.562809, \\ & -0.324014s_1-0.501754s_2-0.719204s_3+0.099562s_4\le -0.052215, \\ & 0.445225s_1-0.346896s_2+0.637939s_3-0.257623s_4\le 0.427920, \\ & -0.202821s_1+0.647361s_2+0.920135s_3-0.983091s_4\le 0.840950, \\ \text{s.t.} & -0.886420s_1-0.802444s_2-0.305441s_3-0.180123s_4\le -1.353686, \\ & -0.515399s_1-0.424820s_2+0.897498s_3+0.187268s_4\le 2.137251, \\ & -0.591515s_1+0.060581s_2-0.427365s_3+0.579388s_4\le -0.290987, \\ & 0.423524s_1+0.940496s_2-0.437944s_3-0.742941s_4\le 0.373620, \\ & s_1\ge 0, s_2\ge 0, s_3\ge 0, s_4\ge 0. \end{cases} \tag{37}$$

Example 4 (see [12])

$$\begin{cases} \min & 6.5s_1-0.5s_1^2-s_2-2s_3-3s_4-2s_5-s_6 \\ & s_1+2s_2+8s_3+s_4+3s_5+5s_6\le 16, \\ & -8s_1-4s_2-2s_3+2s_4+4s_5-s_6\le -1, \\ \text{s.t.} & 2s_1+0.5s_2+0.2s_3-3s_4-s_5-4s_6\le 24, \\ & 0.2s_1+2x_2+0.1s_3-4s_4+2s_5+2s_6\le 12, \\ & -0.1s_1-0.5s_2+2s_3+5s_4-5s_5+3s_6\le 3, \\ & 0\le s_i\le 10,\ i=1,2,\ldots,6. \end{cases} \tag{38}$$

Example 5 (see [12])

$$\begin{cases} \min & 2s_1+3s_2-2s_1^2-2s_2^2+2s_1s_2 \\ & -s_1+s_2\le 1, \\ & s_1-s_2\le 1, \\ \text{s.t.} & -s_1+2s_2\le 3, \\ & 2s_1-s_2\le 3, \\ & 0\le s_1\le 15,\ 0\le s_2\le 15. \end{cases} \tag{39}$$

Example 6 (see [12])

$$\begin{cases} \min & s^TQs+c^Ts \\ \text{s.t.} & As\le b, \\ & s\in S^0=\{0\le s_1\le 2, 0\le s_2\le 2\}, \end{cases} \tag{40}$$

where

$$\begin{aligned} c &= (2,4)^T, \\ b &= (1,2,4,3,1)^T, \\ Q &= \begin{pmatrix} -1 & 2 \\ 2 & -4 \end{pmatrix}, \\ A &= \begin{pmatrix} -4 & 2 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & -4 \end{pmatrix}. \end{aligned} \tag{41}$$

Example 7 (see [26])

$$\begin{cases} \min & (s_1+s_2)(s_1-s_2+7) \\ & 2s_1+s_2 \le 14, \\ & s_1+s_2 \le 10, \\ & -4s_1+s_2 \le 0, \\ \text{s.t.} & 2s_1+s_2 \ge 6, \\ & s_1+2s_2 \ge 6, \\ & s_1-s_2 \le 3, \\ & 1.99 \le s_1 \le 2.01, 7.99 \le s_2 \le 8.01. \end{cases} \quad (42)$$

Example 8 (see [26])

$$\begin{cases} \min & (6s_1+s_2+1)(s_1+2s_2+1)+(-s_1+3)(s_1+s_2+1) \\ & -2s_1+s_2 \le 0, \\ \text{s.t.} & s_1 \le 2.5, \\ & s_1+s_2 \le 8, \\ & s_1, s_2 \ge 0. \end{cases} \quad (43)$$

Example 9 (see [26])

$$\begin{cases} \min & (s_1+2s_2-2)(-2s_1-s_2+3)+(3s_1-2s_2+3)(s_1-s_2-1) \\ & -2s_1+3s_2 \le 6, \\ \text{s.t.} & 4s_1-5s_2 \le 8, \\ & 4s_1-3s_2 \le -12, \\ & s_1, s_2 \ge 0. \end{cases} \quad (44)$$

Example 10

$$\begin{cases} \min & s^T Q s + c^T s + d \\ \text{s.t.} & \mathrm{A}s \le b, \\ & s_i \ge 0,\ i = 1, \ldots, 4, \end{cases} \quad (45)$$

where

$$\begin{aligned} Q &= \begin{pmatrix} 0.2051 & 0.2169 & 0.4600 & 0.2560 \\ 0.2169 & 0.1063 & 0.7901 & 0.4468 \\ 0.4600 & 0.7901 & 0.2844 & 0.1411 \\ 0.2560 & 0.4468 & 0.1411 & 0.0686 \end{pmatrix}, \\ b &= (4.562809, -1.052215, 0.427920, 0.840950, -1.353686, 2.137251, -0.290987, 0.373620)^T, \\ c &= (0.1329, 0.1678, 0.2311, 0.1271)^T, \\ d &= 0.02, \\ A &= \begin{pmatrix} 0.588509 & 0.063565 & 0.945686 & 0.210704 \\ -0.324014 & -0.501754 & -0.719204 & 0.099562 \\ 0.445225 & -0.346896 & 0.637939 & -0.257623 \\ -0.202821 & 0.647361 & 0.920135 & -0.983091 \\ -0.886420 & -0.802444 & -0.305441 & -0.180123 \\ -0.515399 & -0.424820 & 0.897498 & 0.187268 \\ -0.591515 & 0.060581 & -0.427365 & 0.579388 \\ 0.423524 & 0.940496 & -0.437944 & -0.742941 \end{pmatrix}. \end{aligned} \quad (46)$$

Example 11

$$\begin{cases} \min & s^T Q s + c^T s \\ \text{s.t.} & \begin{aligned} & As \le b, \\ & s_i \ge 0,\ i = 1, \ldots, 4, \end{aligned} \end{cases} \tag{47}$$

where

$$\begin{aligned}
Q &= \begin{pmatrix} 0.4296 & 0.3609 & 1.0384 & 0.6327 \\ 0.3609 & 0.1697 & 1.2880 & 0.7429 \\ 1.0384 & 1.2880 & 1.2166 & 0.8716 \\ 0.6327 & 0.7429 & 0.8716 & 0.5973 \end{pmatrix}, \\
b &= (5.562809, -2.052215, 1.427920, 1.840950, -2.353686, 3.137251, -1.290987, 1.373620)^T, \\
c &= (0.4493, 0.3232, 1.2553, 0.7478)^T, \\
d &= 0.1120, \\
A &= \begin{pmatrix} 1.588509 & 0.063565 & 0.945686 & 0.210704 \\ -1.324014 & -0.501754 & -0.719204 & 0.099562 \\ 1.445225 & -0.346896 & 0.637939 & -0.257623 \\ -1.202821 & 0.647361 & 0.920135 & -0.983091 \\ -1.886420 & -0.802444 & -0.305441 & -0.180123 \\ -1.515399 & -0.424820 & 0.897498 & 0.187268 \\ -1.591515 & 0.060581 & -0.427365 & 0.579388 \\ 1.423524 & 0.940496 & -0.437944 & -0.742941 \end{pmatrix}.
\end{aligned} \tag{48}$$

A randomly generated test example 12 with a large scale of variables and constraints is used to validate the robustness and effectiveness of this algorithm; the randomly generated problems are listed as follows and its computational results are listed in Table 2:

Example 12 (see [12])

$$\begin{cases} \min & s^T Q s + c^T s \\ \text{s.t.} & \begin{aligned} & As \le b, \\ & s \in S^0 = \{-10 \le s_i \le 10,\ i = 1, 2, \ldots, n\}, \end{aligned} \end{cases} \tag{49}$$

where Q is an $n \times n$ real symmetric matrix, A is an $m \times n$ real symmetric matrix, $c \in R^n$, and $b \in R^m$; all elements of Q, A, and c are randomly generated from $[-2, 2]$; all elements of b are randomly generated in $[1, 10]$. Obviously, Q may be a positive semidefinite matrix; that is to say, the randomly generated test example 12 may be a nonconvex quadratic programming problem.

For example 12, we solve 10 different random examples with the same parameters and present the numerical results in Table 2. Several notations have been used in column headers of Table 2: Ave.NT denote the average number of iterations; Avg.Time (s) denotes the average CPU execution time of this algorithm in seconds; m denotes the number of constraints; and n denotes the number of variables. In addition, in Table 2, "–" denotes the situation that some of the ten random instances failed to terminate in 3600 s.

It should be noted that, in Tables 1 and 2, since the existing algorithm in the literature [12, 20–24, 26] and the algorithm proposed in this paper use branch-and-bound relaxation structure, we compare our algorithm with the algorithm in the literature [12, 20–24, 26]. In Tables 1 and 2, the software "SCIP" is a commercial solver, which can be used to obtain the global optimal solutions of examples 1-12, so we give the computational comparisons among the proposed algorithm and the commercial solver "SCIP".

From the numerical results of Table 1, we can follow that, for the deterministic examples 1-11, the proposed algorithm in this paper can obtain the same global optimal solution and optimal value as the existing algorithms [12, 20–24, 26] and the commercial solver "SCIP". In addition, when $n \le 3$, the commercial solver "SCIP" has higher computational efficiency with less iterations and computation time. But, when $n \ge 4$, the computational performance of the proposed branch-and-reduction algorithm is significantly higher than that of commercial solver "SCIP".

From the numerical results of Table 2, with any given convergent error, we can follow that the proposed branch-and-reduction algorithm can be used to globally solve the IQPP with a large scale of constraints and variables. From Table 2, when m and n are less than 50, it can be seen that the algorithm can find the global optimal solution of the IQPP with a short time and a smaller number of iterations. As the problem size increases, the Ave.NT and the Avg.Time (s) also increase, but they are not very sensitive to the size of the IQPP.

From numerical results in Table 2, for randomly generated large-size test example 12, compared with the algorithm of [12] and the commercial solver "SCIP", the proposed branch-and-reduction algorithm has the higher computational efficiency. Especially with the increase of the scale of example 12, our algorithm is superior to the known commercial solver "SCIP"; this is because the commercial solver "SCIP" failed to terminate in 3600 s, but the proposed algorithm can find the corresponding global optimal solution of each randomly generated large-size test instance.

In all, from the numerical results in Tables 1 and 2, we can also draw the following conclusions: when the number of variables $n \leq 3$, the commercial solver "SCIP" has higher computational efficiency than the present algorithm in this paper. But when the number of variables $n \geq 4$, the present algorithm in this paper has higher computational efficiency than the commercial solver "SCIP." Furthermore, from the numerical results of example 12, when the number of variables $n \geq 5$, the present algorithm in this paper has significantly higher computational efficiency than the commercial solver "SCIP". In addition, with the increase of the scale of the IQPP, the proposed algorithm is superior to the known commercial solver "SCIP" for solving the IQPP. In all, from the numerical results in Tables 1 and 2, when the scale of the problem is small, for example, the number of variables being less than or equal to 2, the commercial solver is more efficient than the proposed algorithm in this paper, but when the number of variables is greater than or equal to 5, the present algorithm has an obvious advantage over the commercial solver "SCIP". Especially when n is greater than or equal to 15, the SCIP solver failed to terminate in 3600*s*. Therefore, the proposed algorithm in this paper highly outperforms the commercial solver "SCIP" in computational performance.

6. Concluding Remarks

In this paper, we present and validate a novel rectangular branch-and-reduction algorithm for the IQPP. First of all, we transform the IQPP into the EBOP; then by utilizing the characters of quadratic function, we can construct a novel linearizing technique, and by utilizing the linearizing technique, we can derive the LRPP of the EBOP, which can be used to compute the lower bound of the global optimal value of the IQPP. For improving the convergent speed of the algorithm, a new rectangular reduction technique is introduced or constructed. And combining the constructed rectangular reduction technique with the LRPP in a branch-and-bound framework, a new rectangular branch-and-reduction algorithm is established. In order to obtain a global optimal solution of the EBOP, the main computational works of the algorithm involve solving a sequence of LRPP. Finally, the global convergent property of the algorithm is derived, and comparing with the known algorithms, numerical computational results demonstrate the higher computational efficiency and the better computational performance of the algorithm.

In the future, the proposed linear relaxation method and the rectangular reduction method can be extended and applied to solve the generalized linear multiplicative programming problem, the integer quadratic programming problem, and the integer generalized linear multiplicative programming problem.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] E. Matskani, N. Sidiropoulos, Z.-Q. Zhi-quan Luo, and L. Tassiulas, "Convex approximation techniques for joint multiuser downlink beamforming and admission control," *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2682–2693, 2008.

[2] I. M. Bomze and M. L. Overton, "Narrowing the difficulty gap for the Celis-Dennis-Tapia problem," *Mathematical Programming*, vol. 151, no. 2, pp. 459–476, 2015.

[3] J. Yin, H. Jiao, and Y. Shang, "Global algorithm for generalized affine multiplicative programming problem," *IEEE Access*, vol. 7, pp. 162245–162253, 2019.

[4] P. Shen and B. Huang, "Global algorithm for solving linear multiplicative programming problems," *Optimization Letters*, vol. 14, no. 3, pp. 693–710, 2020.

[5] P. Shen, Z. Zhu, and X. Chen, "A practicable contraction approach for the sum of the generalized polynomial ratios problem," *European Journal of Operational Research*, vol. 278, no. 1, pp. 36–48, 2019.

[6] Y. Hu, X. Yang, C. Yang, and S. Li, "An efficient quadratic constrained least squares localization method for narrow space with ranging measurement," *IEEE Access*, vol. 7, pp. 174962–174971, 2019.

[7] S.-J. Qu, K.-C. Zhang, and Y. Ji, "A global optimization algorithm using parametric linearization relaxation," *Applied Mathematics and Computation*, vol. 186, no. 1, pp. 763–771, 2007.

[8] F. A. Al-Khayyal, C. Larsen, and T. Van Voorhis, "A relaxation method for nonconvex quadratically constrained quadratic programs," *Journal of Global Optimization*, vol. 6, no. 3, pp. 215–230, 1995.

[9] H. Jiao, S. Liu, and N. Lu, "A parametric linear relaxation algorithm for globally solving nonconvex quadratic programming," *Applied Mathematics and Computation*, vol. 250, pp. 973–985, 2015.

[10] Y. Gao, H. Xue, and P. Shen, "A new rectangle branch-and-reduce approach for solving nonconvex quadratic programming problems," *Applied Mathematics and Computation*, vol. 168, no. 2, pp. 1409–1418, 2005.

[11] Y. Gao, Y.-l. Shang, and L. Zhang, "A branch and reduce approach for solving nonconvex quadratic programming

problems with quadratic constraints," *OR Transactions*, vol. 9, no. 2, pp. 9–20, 2005.

[12] L. Ge and S. Liu, "An accelerating algorithm for globally solving nonconvex quadratic programming," *Journal of Inequalities and Applications volume*, vol. 2018, p. 178, 2018.

[13] H. Jiao, Z. Wang, and Y. Chen, "Global optimization algorithm for sum of generalized polynomial ratios problem," *Applied Mathematical Modelling*, vol. 37, no. 1-2, pp. 187–197, 2013.

[14] H. Jiao and S. Liu, "An efficient algorithm for quadratic sum-of-ratios fractional programs problem," *Numerical Functional Analysis and Optimization*, vol. 38, no. 11, pp. 1426–1445, 2017.

[15] H. Jiao, Y. Chen, and W. Cheng, "A novel optimization method for nonconvex quadratically constrained quadratic programs," *Abstract and Applied Analysis*, vol. 2014, Article ID 698489, 11 pages, 2014.

[16] H. Jiao and S. Liu, "Range division and compression algorithm for quadratically constrained sum of quadratic ratios," *Computational and Applied Mathematics*, vol. 36, no. 1, pp. 225–247, 2017.

[17] H. Jiao and R. Chen, "A parametric linearizing approach for quadratically inequality constrained quadratic programs," *Open Mathematics*, vol. 16, no. 1, pp. 407–419, 2018.

[18] Z. Hou, H. Jiao, L. Cai, and C. Bai, "Branch-delete-bound algorithm for globally solving quadratically constrained quadratic programs," *Open Mathematics*, vol. 15, no. 1, pp. 1212–1224, 2017.

[19] D. Vandenbussche and G. L. Nemhauser, "A branch-and-cut algorithm for nonconvex quadratic programs with box constraints," *Mathematical Programming*, vol. 102, no. 3, pp. 559–575, 2005.

[20] Y. Chen and H. Jiao, "A nonisolated optimal solution of general linear multiplicative programming problems," *Computers and Operations Research*, vol. 36, no. 9, pp. 2573–2579, 2009.

[21] Y. Gao, G. Wu, and W. Ma, "A new global optimization approach for convex multiplicative programming," *Applied Mathematics and Computation*, vol. 216, no. 4, pp. 1206–1218, 2010.

[22] C.-F. Wang, S.-Y. Liu, and P.-P. Shen, "Global minimization of a generalized linear multiplicative programming," *Applied Mathematical Modelling*, vol. 36, no. 6, pp. 2446–2451, 2012.

[23] H.-W. Jiao, S.-Y. Liu, and Y.-F. Zhao, "Effective algorithm for solving the generalized linear multiplicative problem with generalized polynomial constraints," *Applied Mathematical Modelling*, vol. 39, no. 23-24, pp. 7568–7582, 2015.

[24] N. V. Thoai, "A global optimization approach for solving the convex multiplicative programming problem," *Journal of Global Optimization*, vol. 1, no. 4, pp. 341–357, 1991.

[25] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, Springer Science and Business Media, Berlin, Germany, 2013.

[26] Y. Zhao, *Global Optimization Study Research on Two Classes of Multiplicative Programming Problems*, Master's Thesis, Henan Normal University, Henan, China, 2014.

[27] P. Bonami, A. Lodi, J. Schweiger, and A. Tramontani, "Solving quadratic programming by cutting planes," *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 1076–1105, 2019.

[28] C.-F. Wang, Y.-Q. Bai, and P.-P. Shen, "A practicable branch-and-bound algorithm for globally solving linear multiplicative programming," *Optimization*, vol. 66, no. 3, pp. 397–405, 2017.

[29] H. Jiao, W. Wang, R. Chen, Y. Shang, and J. Yin, "An efficient outer space algorithm for generalized linear multiplicative programming problem," *IEEE Access*, vol. 8, p. 80629. in press, 2020.

[30] P. Shen, B. Huang, and L. Wang, "Range division and linearization algorithm for a class of linear ratios optimization problems," *Journal of Computational and Applied Mathematics*, vol. 350, pp. 324–342, 2019.

[31] P. P. Shen, T. L. Zhang, and C. F. Wang, "Solving a class of generalized fractional programming problems using the feasibility of linear programs," *Journal of Inequalities and Applications*, vol. 147, 2017.

[32] P. Shen and C. Wang, "Linear decomposition approach for a class of nonconvex programming problems," *Journal of Inequalities and Applications*, vol. 2017, no. 1, 2017.

[33] S. Liu and Y. Zhao, "An efficient algorithm for globally solving generalized linear multiplicative programming," *Journal of Computational and Applied Mathematics*, vol. 296, pp. 840–847, 2016.

[34] R. M. Fukuda and T. Abrao, "Linear, quadratic, and semidefinite programming massive MIMO detectors: reliability and complexity," *IEEE Access*, vol. 7, pp. 29506–29519, 2019.

[35] Y. Pei and D. Zhu, "Local convergence of a trust-region algorithm with line search filter technique for nonlinear constrained optimization," *Applied Mathematics and Computation*, vol. 273, pp. 797–808, 2016.

[36] D. Liao-McPherson and I. Kolmanovsky, "FBstab: a proximally stabilized semismooth algorithm for convex quadratic programming," *Automatica*, vol. 113, Article ID 108801, 2020.

[37] H. Sheen and M. Yamashita, "Exploiting aggregate sparsity in second-order cone relaxations for quadratic constrained quadratic programming problems," *Optimization Methods and Software*, p. 1, 2020.

[38] M. A. Elsisy, D. A. Hammad, and M. A. El-Shorbagy, "Solving interval quadratic programming problems by using the numerical method and swarm algorithms," *Complexity*, vol. 2020, Article ID 6105952, 11 pages, 2020.

[39] X. Leng, S. Piao, L. Chang, Z. He, and Z. Zhu, "Universal walking control framework of biped robot based on dynamic model and quadratic programming," *Complexity*, vol. 2020, Article ID 2789039, 13 pages, 2020.

[40] R. Fangnon, C. Ainamon, A. V. Monwanou, C. H. Miwadinou, and J. B. Chabi Orou, "Nonlinear dynamics of the quadratic-damping helmholtz oscillator," *Complexity*, vol. 2020, Article ID 8822534, 17 pages, 2020.

[41] X. Liu, Y. L. Gao, B. Zhang, and F. P. Tian, "A new global optimization algorithm for a class of linear fractional programming," *Mathematics*, vol. 7, no. 9, p. 867, 2019.