Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 16, No. 3, pp. 441–459, 2020

A COMBINATION OF LINEAR APPROXIMATION AND LAGRANGIAN DUAL WITH A SIMPLE CUT FOR GENERAL SEPARABLE INTEGER PROGRAMMING PROBLEMS

Fenlan Wang

Abstract: In this paper, a new simple and exact algorithm is presented for general separable integer programming problems. The algorithm incorporates a special domain cut into the branch and bound method. The domain cut technique removes some regions that don't contain optimal solutions to the primal problem from the hyper-rectangle, which makes the presented algorithm in this paper different from the traditional branch and bound method. Also the duality gap can be reduced in the domain cut process. The lower bound can be calculated easily by solving a linear programming problem and its Lagrangian dual problem. The computational experiments are reported for two kinds of convex, concave or nonconvex and nonconcave objective functions respectively in the paper. The numerical comparison with the traditional branch and bound method for a quadratic polynomial convex objective function is given and the comparison results show the algorithm is encouraging.

Key words: *Separable integer programming, linear approximation, Lagrangian dual, branch and bound, duality gap*

Mathematics Subject Classification: *90C10, 90C26, 90C30*

1 Introduction

In this paper the following separable integer programming problems are considered:

$$
\begin{aligned}
(P) \qquad & \min\ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.}\ \ Ax \le b, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l \le x \le u\},
\end{aligned}
$$

where $f_j(x_j) \in \mathbb{C}^2, j = 1, \dots, n$ can be convex, concave or nonconvex and nonconcave functions, and $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\mathbb{Z}^n$ is the set of all integer points in $\mathbb{R}^n$. Further $l = (l_1, l_2, \dots, l_n)^T$ and $u = (u_1, u_2, \dots, u_n)^T \in \mathbb{Z}^n$ are the lower and upper bounds of the variable x, respectively.

Nonlinear separable integer programming models have many applications in engineering such as capital budgeting [21], capacity planning [6], optimization problems from graph theory [1, 17], fixed charges problems with integer variables [13] and problems involving economies of scale. Also there are many methods in literatures for solving nonlinear separable integer programming problems. Most of them are dynamic programming-based methods and continuous relaxation-based branch and bound methods or a combination of the dynamic programming method and the branch and bound method. The dynamic programming method is used to solve separable integer programming problems with a single constraint [2,10,14]. Due to the property 'curse of dimensionality' of the dynamic programming, it is difficult to be extended to solve multiple constrained separable integer programming problems. Branch and bound methods based on the continuous relaxation problem are used for solving convex integer programming problems, since the continuous relaxation problems can be solved easily [4, 7, 8, 11, 15, 16, 21, 24, 25]. For concave integer programming problems, branch and bound methods based on global optimization over a polyhedron were presented in [3, 5, 9, 12]. Hybrid approach that is a combination of the dynamic programming method and the branch-and-bound method was presented in [20, 22, 23] for solving nonlinear separable integer programming problems. Although this method partially overcomes the curse of dimensionality of the dynamic programming, it requires the objective function to be nonincreasing and needs to store many feasible solutions that may cause a storage problem. A new exact algorithm is presented in [18] for separable quadratic integer programming problems and the numerical results were also reported. This algorithm adopts the contour cut technique according to the properties of the objective function. Recently, a new domain cut technique is presented in [26] for solving separable integer programming problems with a concave objective function and linear constraints, and the numerical results were also given.

In this paper, we extend the algorithm in [26] to solve general separable integer programming problem with linear constraints. The objective function can be convex, concave or nonconvex and nonconcave functions. The proposed algorithm is essentially a frame of the branch and bound method, but it is very different from the traditional branch and bound method. The lower bound of the presented algorithm is obtained by combining a simple linear programming problem with the Lagrangian dual problem. As we know, the Lagrangian dual method provides a very efficient way for finding a lower bound of the optimal objective function value of separable integer programming problems, since the Lagrangian relaxation problem can be solved easily by being decomposed into n subproblems of minimizing a one-dimensional nonlinear function over integers in a finite interval. Thus the dual optimal solution can be searched for efficiently. The optimal dual value is a lower bound of the optimal objective function value of problem (P). The branches are divided by a special domain cut and partition technique which reduces the feasible region and the duality gap. Thus the optimal solutions can be found quickly in a finite number of iterations. The computational experiments and comparison results are also reported in the paper.

In [19], a convergent Lagrangian and domain cut method is presented for solving separable nonlinear integer programming problems. Compared the algorithm presented in this paper with the method in [19], on the one hand, the lower bound of the problem in [19] is obtained only by solving the Lagrangian dual problem, while we can obtain a better lower bound by solving the linear underestimation problem and the Lagrangian dual problem in this paper. On the other hand, the domain cut methods are also different. The domain cut in [19] depends on monotonicity of the problem where the objective function and the constraint functions are all nondecreasing. But there are no restriction on monotonicity of the problem in this paper.

The remainder of this paper is organized as follows: Section 2 gives the lower bounds for the subproblems with different objective functions: convex, concave or nonconvex and nonconcave. A special domain cut and partition technique is presented in section 3. In section 4, the main algorithm is described in details. Finally, numerical computational and comparison results are reported in section 5.

2 The Lower Bounds for the Subproblems

First some notations are introduced. Denote $[\alpha, \beta]$ as the box (hyper-rectangle) formed by α, β and $\langle \alpha, \beta \rangle$ as the set of integer points in $[\alpha, \beta]$.

$$[\alpha, \beta] = \{x \mid \alpha_j \le x_j \le \beta_j, j = 1, \ldots, n\}$$

$$\langle \alpha, \beta \rangle = \{x \mid \alpha_j \le x_j \le \beta_j,\ x_j \text{ integer},\ j = 1, \ldots, n\} = \Pi_{j=1}^{n} \langle \alpha_j, \beta_j \rangle. \tag{2.1}$$

where $\alpha,\ \beta \in \mathbb{Z}^n$. For convenience, the set $\langle \alpha, \beta \rangle$ is called an integer box and define $[\alpha, \beta] = \langle \alpha, \beta \rangle = \emptyset$ if $\alpha \not\le \beta$. In addition, denote $v(\cdot)$ as the optimal value of the problem $(\cdot)$. Let (SP) be a subproblem of (P) by replacing X with $\langle \alpha, \beta \rangle$ where $l \le \alpha \le \beta \le u$.

Now consider the lower bound of the following subproblem (SP) of (P):

$$(SP) \qquad \begin{aligned} \min\ & f(x) = \sum_{j=1}^{n} f_j(x_j) \\ \text{s.t.}\ & Ax \le b, \\ & x \in \langle \alpha, \beta \rangle. \end{aligned}$$

2.1 Linear underestimation

- $f_j(x_j)$ is a convex function.

  Since $f_j(x_j)$ is a convex function over the interval $[\alpha_j, \beta_j]$, we have $f_j(x_j) \ge f_j(x^0) + f_j'(x^0)(x - x^0)$ where $x^0 \in [\alpha_j, \beta_j]$. Therefore the tangent underestimating function $L_j^1(x_j)$ can be taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$. The linear underestimating function of $f(x) = \sum_{j=1}^{n} f_j(x_j)$ over box $[\alpha, \beta]$ can be expressed as:

$$L(x) = \sum_{j=1}^{n} L_j^1(x_j)$$

  where

$$L_j^1(x_j) = \begin{cases} f_j(x_j^0) + f_j'(x_j^0)(x_j - x_j^0), & \alpha_j < \beta_j, \\ f_j(\alpha_j), & \alpha_j = \beta_j. \end{cases} \tag{2.2}$$

  where x_j^0 is taken as the center point of $[\alpha_j, \beta_j]$, $j = 1, \ldots, n$.

- $f_j(x_j)$ is a concave function.

  The lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$ can be taken as the linear underestimation function of $f_j(x_j)$ over $[\alpha_j, \beta_j]$, which is a line segment connecting two endpoints α_j and β_j of the closed interval $[\alpha_j, \beta_j]$. The linear underestimating function of $f(x) = \sum_{j=1}^{n} f_j(x_j)$ over the box $[\alpha, \beta]$ can be written as:

$$L(x) = \sum_{j=1}^{n} L_j^2(x_j),$$

  where

$$L_j^2(x_j) = \begin{cases} f_j(\alpha_j) + \frac{f_j(\beta_j) - f_j(\alpha_j)}{\beta_j - \alpha_j}(x_j - \alpha_j), & \alpha_j < \beta_j, \\ f_j(\alpha_j), & \alpha_j = \beta_j. \end{cases} \tag{2.3}$$

According to the above two situations, the linear underestimation taken as the lower bound can be extended to the situation where $f_j(x_j)$ is a nonconvex and nonconcave function.

- $f_j(x_j)$ is a nonconvex and nonconcave function.

  When $f_j(x_j)$ is nonconvex and nonconcave, these points satisfying $f_j''(x_j) = 0$ can divide the interval $[\alpha_j, \beta_j]$ into several subintervals, then $f_j(x_j)$ is either convex or concave over these subintervals. Thus the lower bound of $f_j(x_j)$ over each subinterval can be taken as the tangent underestimating function $L_j^1(x_j)$ (2.2) or the linear underestimation function $L_j^2(x_j)$ (2.3).

  In the following, quadratic, cubic or quartic nonconvex and nonconcave objective function $f(x)$ is taken respectively as examples to explain the lower bounds of the subproblems in details, and note that here $f_j(x_j)$ can be convex, concave or nonconvex and nonconcave.

  (i) $f_j(x_j)$ is a quadratic function, i.e. $f_j(x_j) = a_j x_j^2 + b_j x_j, \quad j = 1, 2, \ldots, n.$

  - when $a_j > 0$, $f_j(x_j)$ is a convex function. Thus the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$ can be taken as $L_j^1(x_j)$ (2.2).
  - when $a_j < 0$, $f_j(x_j)$ is a concave function. Then the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$ can be taken as $L_j^2(x_j)$ (2.3).

  (ii) $f_j(x_j)$ is a cubic function, i.e. $f_j(x_j) = a_j x_j^3 + b_j x_j^2 + c_j x_j$. We know $f_j'(x_j) = 3a_j x_j^2 + 2b_j x_j + c_j$ and $f_j''(x_j) = 6a_j x_j + 2b_j$. Let $f_j''(x_j) = 0$, we have $\hat{x}_j = -\frac{b_j}{3a_j}$.

  - when $a_j > 0$,
    - if $\hat{x}_j \leqslant \alpha_j$, then $f_j(x_j)$ must be convex, and $L_j^1(x_j)$ (2.2) is taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$.
    - if $\hat{x}_j \geqslant \beta_j$, then $f_j(x_j)$ must be concave, and $L_j^2(x_j)$ (2.3) is taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$.
    - if $\alpha_j < \hat{x}_j < \beta_j$, then $f_j(x_j)$ is concave over $[\alpha_j, \hat{x}_j]$ and convex over $[\hat{x}_j, \beta_j]$. Thus $f_j(x_j)$ has different convexity and concavity over the closed interval $[\alpha_j, \beta_j]$. In order to easily calculate the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$ via (2.2) and (2.3), we must ensure $f_j(x_j)$ has the same convexity or concavity over an interval. So the interval $[\alpha_j, \beta_j]$ should be partitioned into the union of two subintervals $[\alpha_j, \hat{x}_j]$ and $[\hat{x}_j, \beta_j]$ over which $f_j(x_j)$ is either convex or concave. Thus the lower bound of $f_j(x_j)$ over $[\alpha_j, \hat{x}_j]$ can be taken as the linear underestimation function (2.3) where β_j is replaced by $\hat{x}_j$ and the lower bound of $f_j(x_j)$ over $[\hat{x}_j, \beta_j]$ can be taken as the tangent underestimating function (2.2) where α_j is replaced by $\hat{x}_j$.
  - when $a_j < 0$,
    - if $\hat{x}_j \leqslant \alpha_j$, then $f_j(x_j)$ must be concave and the linear underestimation function (2.3) is taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$.

* if $\hat{x}_j \geqslant \beta_j$, then $f_j(x_j)$ must be convex and the tangent underestimating function (2.2) is taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$.
* if $\alpha_j < \hat{x_j} < \beta_j$, then $f_j(x_j)$ is convex over $[\alpha_j, \hat{x}_j]$ and concave over $[\hat{x}_j, \beta_j]$. Also the closed interval $[\alpha_j, \beta_j]$ should be partitioned into the union of two subintervals $[\alpha_j, \hat{x}_j]$ and $[\hat{x}_j, \beta_j]$ over which $f_j(x_j)$ should be ensured the same convexity or concavity. Thus the lower bound of $f_j(x_j)$ over $[\alpha_j, \hat{x_j}]$ can be taken as the tangent underestimating function (2.2) with β_j replaced by $\hat{x}_j$ and the lower bound of $f_j(x_j)$ over $[\hat{x}_j, \beta_j]$ can be taken as the linear underestimation function (2.3) with α_j replaced by $\hat{x_j}$.

(iii) $f_j(x_j)$ is a quartic function, i.e. $f_j(x_j) = a_j x_j^4 + b_j x_j^3 + c_j x_j^2 + d_j x_j$. We know $f_j'(x_j) = 4a_j x_j^3 + 3b_j x_j^2 + 2c_j x_j + d_j$ and $f_j''(x_j) = 12a_j x_j^2 + 6b_j x_j + 2c_j$. Let $\Delta_j = 9b_j^2 - 24a_j c_j$ and $f_j''(x_j) = 0$, we have $\hat{x}^{(1)_j} = \frac{-3b_j - \sqrt{\Delta_j}}{12a_j}$ and $\hat{x_j^{(2)}} = \frac{-3b_j + \sqrt{\Delta_j}}{12a_j}$.

- $\Delta_j > 0$,

  * when $a_j > 0$,

    · if $\beta_j \leq \hat{x}_j^{(1)}$ or $\alpha_j \geq \hat{x}_j^{(2)}$, then $f_j(x_j)$ must be convex over $[\alpha_j, \beta_j]$.
    · if $\hat{x}_j^{(1)} \leq \alpha_j < \beta_j \leq \hat{x}_j^{(2)}$, then $f_j(x_j)$ must be concave over $[\alpha_j, \beta_j]$.
    · if $\alpha_j < \hat{x}_j^{(1)} < \beta_j < \hat{x}_j^{(2)}$, then $f_j(x_j)$ is convex over $[\alpha_j, \hat{x}_j^{(1)}]$ and concave over $[\hat{x}_j^{(1)}, \beta_j]$.
    · if $\hat{x}_j^{(1)} < \alpha_j < \hat{x}_j^{(2)} < \beta_j$, then $f_j(x_j)$ is concave over $[\alpha_j, \hat{x}_j^{(2)}]$ and convex over $[\hat{x}_j^{(2)}, \beta_j]$.
    · if $\alpha_j < \hat{x}_j^{(1)} < \hat{x}_j^{(2)} < \beta_j$, then $f_j(x_j)$ is convex over $[\alpha_j, \hat{x}_j^{(1)}]$ and $[\hat{x}_j^{(2)}, \beta_j]$, concave over $[\hat{x}_j^{(1)}, \hat{x}_j^{(2)}]$. .

  * when $a_j < 0$,

    · if $\beta_j \leq \hat{x}_j^{(2)}$ or $\alpha_j \geq \hat{x}_j^{(1)}$, then $f_j(x_j)$ must be concave over $[\alpha_j, \beta_j]$.
    · if $\hat{x}_j^{(2)} \leq \alpha_j < \beta_j \leq \hat{x}_j^{(1)}$, then $f_j(x_j)$ must be convex over $[\alpha_j, \beta_j]$.
    · if $\alpha_j < \hat{x}_j^{(2)} < \beta_j < \hat{x}_j^{(1)}$, then $f_j(x_j)$ is concave over $[\alpha_j, \hat{x}_j^{(2)}]$ and convex over $[\hat{x}_j^{(2)}, \beta_j]$.
    · if $\hat{x}_j^{(2)} < \alpha_j < \hat{x}_j^{(1)} < \beta_j$, then $f_j(x_j)$ is convex over $[\alpha_j, \hat{x}_j^{(1)}]$ and concave over $[\hat{x}_j^{(1)}, \beta_j]$.
    · if $\alpha_j < \hat{x}_j^{(2)} < \hat{x}_j^{(1)} < \beta_j$, then $f_j(x_j)$ is concave over $[\alpha_j, \hat{x}_j^{(2)}]$ and $[\hat{x}_j^{(1)}, \beta_j]$, convex over $[\hat{x}_j^{(2)}, \hat{x}_j^{(1)}]$.

- $\Delta_j \leq 0$,

* when $a_j > 0$, $f''_j(x_j) \geq 0$, so $f_j(x_j)$ is always convex over $[\alpha_j, \beta_j]$.
* when $a_j < 0$, $f''_j(x_j) \leq 0$, so $f_j(x_j)$ is always concave over $[\alpha_j, \beta_j]$.

As discussed above, if $f_j(x_j)$ is always convex over $[\alpha_j, \beta_j]$, the tangent underestimating function $L_j^1(x_j)$ (2.2) is taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$; if $f_j(x_j)$ is always concave over $[\alpha_j, \beta_j]$, the linear underestimation function $L_j^2(x_j)$ (2.3) is taken as the lower bound of $f_j(x_j)$ over $[\alpha_j, \beta_j]$; if $f_j(x_j)$ has different convexity and concavity over $[\alpha_j, \beta_j]$, i.e., $f_j(x_j)$ is convex over one subinterval $[\alpha_j, \gamma]$ and $f_j(x_j)$ is concave over another subinterval $[\gamma, \beta_j]$, then the closed interval $[\alpha_j, \beta_j]$ should be divided into two subintervals $[\alpha_j, \gamma]$ and $[\gamma, \beta_j]$, so that $f_j(x_j)$ has the same convexity or concavity over each subinterval, and the lower bound of $f_j(x_j)$ can be calculated easily via (2.2) or (2.3). Of course the integer box $\langle \alpha, \beta \rangle$ should also be accordingly partitioned into the union of some integer subboxes, over which all $f_j(x_j)$, $j = 1, 2, \ldots$,n, should be ensured to be either convex or concave.

Thus the linear approximation problem of (SP) is as follows:

$$(LP) \qquad \min \ L(x) = \sum_{j=1}^{n} L_j(x_j)$$
$$\text{s.t.} \ \ Ax \leq b,$$
$$x \in [\alpha, \beta],$$

where $L_j(x_j)$ is $L_j^1(x_j)$ or $L_j^2(x_j)$. Obviously, the problem (LP) is a linear programming problem which can be solved easily via the simplex method. Also $v(LP)$ is the lower bound of the problem (SP).

2.2 Lagrangian duality and dual search

As we know, the Lagrangian dual method to find the lower bound is a very efficient method for separable integer programming problems. Therefore the Lagrangian dual method can provide us another lower bound for (SP).

The Lagrangian relaxation of (SP) is

$$(L_\mu) \quad d(\mu) = \min_{x \in \langle \alpha, \beta \rangle} L(x, \mu)$$

where

$$L(x, \mu) = f(x) + \mu^T (Ax - b), \qquad \mu \geq 0$$

Let

$$S = \{x \in \langle \alpha, \beta \rangle | Ax \leq b\},$$
$$f^* = \min_{x \in S} f(x)$$

Then the following weak duality holds

$$d(\mu) \leq f(x), \quad \forall x \in S, \quad \mu \geq 0.$$

Therefore, $d(\mu)$ always provides a lower bound for f^*. The Lagrangian dual problem of (SP) is

$$(D) \qquad \max_{\mu \geq 0} d(\mu).$$

Let μ^* be the optimal solution to (D). The nonnegative constant $f^* - d(\mu^*)$ is called the duality gap of the problem.

Due to the separability of $f(x)$, the problem (L_μ) can be calculated very easily over $\langle \alpha, \beta \rangle$ via decomposition:

$$\begin{aligned} L(x,\mu) &= f(x) + \mu^T(Ax-b) \\ &= \sum_{j=1}^{n} f_j(x_j) + \mu^T(Ax-b) \\ &= -\mu^T b + \sum_{j=1}^{n}(f_j(x_j) + \mu^T a_j x_j) \end{aligned}$$

where $a_j,\ j = 1, 2, \dots, n$ is column vectors of matrix A. That is $A = (a_1, a_2, \dots, a_n)$.

The subgradient method can be used to update the Lagrangian multiplier vector μ for the problem (D):

$$\mu_i^{k+1} = \max\{0, \mu_i^k - t^k h_i^k / \|h^k\|\}, \qquad i = 1, \dots, m, \quad k = 1, \dots,$$

where $h^k = Ax - b$ and t^k is the stepsize satisfying the conditions:

$$t^k \to 0, \quad \sum_{k=1}^{\infty} t^k = +\infty.$$

We can take $t^k = \frac{1}{2k}$. Due to the slow convergence of the subgradient method, there is a tradeoff between CPU time and the accuracy of the solution. So we can stop at an approximate optimal solution either when $\|\mu^{k+1} - \mu^k\|$ is small enough or the number of iterations exceeds a given maximum iteration number. The dual search procedure will be described as follows.

Procedure 2.1 (Lagrangian dual search).

Step 0. Let $\mu^1 = 1, v = -1.d + 10$ and $k = 1$.

Step 1. If $k > M$ (M is a given maximum iteration number), then stop and v is the approximate optimal value of (D). Otherwise, go to Step 2.

Step 2. Solve (L_{μ^k}) and yield an optimal solution x^k with the optimal value $d(\mu^k)$.

Step 3. If $|d(\mu^k) - v| < \epsilon$, then stop and v is the approximate optimal value of (D). Otherwise go to Step 4.

Step 4. $h^k = Ax^k - b$, $t^k = \frac{1}{2k}$, $\mu_i^{k+1} = \max\{0, \mu_i^k - t^k h_i^k/\|h^k\|\},\ i = 1, \dots, m$. If $\|\mu^{k+1} - \mu^k\| < \epsilon$, then stop and v is the approximate optimal value of (D). Otherwise, let $v := \max\{v, d(\mu^k)\}$ and $k := k + 1$, then go to Step 1.

Thus the lower bound of the problem (SP) over the integer subbox $\langle \alpha, \beta \rangle$ can be taken as $\max\{v(LP), v(D)\}$.

3 Domain Cut and Partition

This section will describe a special domain cut technique which cut off some integer subboxes that do not contain the optimal integer solution of (P) from the super-rectangle domain.

By solving the problem (LP), we obtain a continuous feasible solution $\tilde{x}$ and a lower bound $v(LP)$ of (SP) in the integer subbox $\langle\alpha,\beta\rangle$. The following two cases need to be considered.

Case (a): If $\tilde{x}$ is an integer solution, obviously $f(\tilde{x})$ is an upper bound of (P).

- For the convex objective function, the following lemma will give us a special domain cut technique about $\tilde{x}$:

Lemma 3.1. *there is no feasible solution better than $\tilde{x}$ in the integer subbox $R(\tilde{x}) = \langle\bar{\alpha},\bar{\beta}\rangle$, where $\bar{\alpha},\bar{\beta}$ are defined as follows:*

$$\bar{\alpha}_j = \begin{cases} \tilde{x}_j, & (\nabla f(\tilde{x}))_j > 0 \\ \alpha_j, & (\nabla f(\tilde{x}))_j < 0 \end{cases} \tag{3.1}$$

$$\bar{\beta}_j = \begin{cases} \beta_j, & (\nabla f(\tilde{x}))_j > 0 \\ \tilde{x}_j, & (\nabla f(\tilde{x}))_j < 0 \end{cases}$$

Proof. Since the objective function $f(x)$ is a convex function, it is bounded below by the hyperplane $g(x) = f(\tilde{x}) + (\nabla f(\tilde{x}))^T(x-\tilde{x})$. By (3.1), for all $x \in R(\tilde{x})$ we have $(\nabla f(\tilde{x}))^T(x-\tilde{x}) \geq 0$ Thus $f(x) \geq g(x) \geq f(\tilde{x})$. □

Then the integer box $R(\tilde{x})$ can be cut off from $\langle\alpha,\beta\rangle$ without remove any feasible solutions better than $\tilde{x}$.

- For the concave objective function,

  - If $f_j(x_j), j = 1,\ldots,n$ are quadratic concave functions, consider the ellipsoid contour of $f(x)$:

  $$\sum_{i=1}^{n}[c_j x_j^2 + d_j x_j] = f(\tilde{x})$$

  where $c_j < 0, j = 1,\ldots,n$. The center of the ellipsoid is $o = (o_1, o_2, \ldots, o_n) = (-\frac{d_1}{2c_1}, -\frac{d_2}{2c_2}, \ldots, -\frac{d_n}{2c_n})^T$. By the symmetry of the ellipsoid contour, the maximum integer subbox $\langle\bar{\alpha},\bar{\beta}\rangle$ inside the ellipsoid passing through $\tilde{x}$ can be found, where

  $$\begin{aligned} \bar{\alpha} &= (\lceil o_1 - |\tilde{x}_1 - o_1|\rceil, \ldots, \lceil o_n - |\tilde{x}_n - o_n|\rceil), \\ \bar{\beta} &= (\lfloor o_1 + |\tilde{x}_1 - o_1|\rfloor, \ldots, \lfloor o_n + |\tilde{x}_n - o_n|\rfloor). \end{aligned} \tag{3.2}$$

  Then we can conclude that the domain $\langle\bar{\alpha},\bar{\beta}\rangle \cap \langle\alpha,\beta\rangle$ can be cut off from $\langle\alpha,\beta\rangle$ and will not remove any feasible solutions better than $\tilde{x}$.

  - If $f_j(x_j), j = 1,\ldots,n$ are not quadratic concave functions, we can at least cut the point $\{\tilde{x}\}$ off from $\langle\alpha,\beta\rangle$. That is $\langle\bar{\alpha},\bar{\beta}\rangle = \{\tilde{x}\}$.

- For the indefinite objective function, we can also cut the point $\{\tilde{x}\}$ off from $\langle\alpha,\beta\rangle$. That is $\langle\bar{\alpha},\bar{\beta}\rangle = \{\tilde{x}\}$.

Case (b): If $\tilde{x}$ is not an integer solution, then we can obtain two integer points $x^{(1)}$ and $x^{(2)}$ by rounding $\tilde{x}$ up or down along the gradient direction $\nabla L(\tilde{x})$ and the negative gradient direction $-\nabla L(\tilde{x})$ of (LP) respectively. Also the following lemmas will present us the special domain cut about $x^{(1)}$ and $x^{(2)}$:

Lemma 3.2. *$x^{(2)}$ must be an infeasible solution, there is no feasible solution in the integer box $N_1(x^{(2)}) = \langle \gamma, \delta \rangle$, where γ, δ are determined by*

$$\gamma_j = \begin{cases} (x^{(2)})_j, & (\nabla L(x^{(2)}))_j < 0 \\ \alpha_j, & (\nabla L(x^{(2)}))_j > 0 \end{cases} \tag{3.3}$$

$$\delta_j = \begin{cases} \beta_j, & (\nabla L(x^{(2)}))_j < 0 \\ (x^{(2)})_j, & (\nabla L(x^{(2)}))_j > 0 \end{cases}$$

Proof. Suppose $x^{(2)}$ is a feasible solution. Since $x^{(2)}$ is obtained by rounding $\tilde{x}$ up or down along the negative gradient direction $-\nabla L(\tilde{x})$ of the problem (LP), we must have $L(x^{(2)}) < L(\tilde{x})$, which is in contradiction with $\tilde{x}$ being the optimal solution to the problem (LP).

$N_1(x^{(2)})$ is from $x^{(2)}$ along the negative gradient direction $-\nabla L(x^{(2)})$ which is a descent direction and the feasible region is a convex set, so there must be not any feasible solutions in the integer box $N_1(x^{(2)})$. □

The integer box $N_1(x^{(2)})$ can also be discarded from $\langle \alpha, \beta \rangle$ without removing any feasible solutions.

For $x^{(1)}$, two cases are considered according to whether $x^{(1)}$ being feasible or not.

- If $x^{(1)}$ is feasible, we can cut off the integer box $R(x^{(1)}) = \langle \bar{\alpha}, \bar{\beta} \rangle$ accordingly as the above Case (a) where $\tilde{x}$ is replaced by $x^{(1)}$.
- If $x^{(1)}$ is not a feasible solution, without loss of generality, suppose $A_i x^{(1)} > b_i$, where $A_i = (a_{i1}, a_{i2}, \ldots, a_{in})$. Then we have the following lemma.

Lemma 3.3. *There is no feasible solution in the integer box $N_2(x^{(1)}) = \langle \gamma, \delta \rangle$, where γ, δ are determined by*

$$\gamma_j = \begin{cases} (x^{(1)})_j, & (A_i)_j > 0 \\ \alpha_j, & (A_i)_j < 0 \end{cases} \tag{3.4}$$

$$\delta_j = \begin{cases} \beta_j, & (A_i)_j > 0 \\ (x^{(1)})_j, & (A_i)_j < 0 \end{cases}$$

Proof. For any $x \in N_2(x^{(1)})$, we have $A_i(x - x^{(1)}) > 0$ by (3.4). So $A_i x > A_i x^{(1)} > b_i$. Thus there is no feasible solution in this integer box $N_2(x^{(1)})$. □

Cut the integer box $N_2(x^{(1)})$ off from $\langle \alpha, \beta \rangle$ without missing any feasible solutions.

The above lemmas show us the special domain cut skills which cut off some integer subboxes not containing the optimal solution to the problem (P) from a hyper-rectangle. Thus the feasible region is reduced greatly and the optimal solution to the problem (P) can be found more quickly.

After the domain cut, the revised domain $(\langle \alpha, \beta \rangle \backslash \langle \bar{\alpha}, \bar{\beta} \rangle) \backslash \langle \gamma, \delta \rangle$ usually isn't an integer box. In order to easily calculate the lower bound of the problem (P) over the revised domain by solving a linear programming problem (LP) in Section 2, the revised domain need to be divided into a union of some integer subboxes, over which new subproblems are generated. The following lemma will show us how to divide the revised domain into a union of some integer subboxes.

Lemma 3.4. *(see [18]) Let α, β, γ, $\delta \in \mathbb{Z}^n$ and $\langle\alpha,\beta\rangle$, $\langle\gamma,\delta\rangle$ be two integer subboxes satisfying $\alpha \le \gamma \le \delta \le \beta$. Then*

$$\begin{aligned} \langle\alpha,\beta\rangle \setminus \langle\gamma,\delta\rangle \quad = \quad & \{\cup_{j=1}^{n} \left(\Pi_{i=1}^{j-1}\langle\alpha_i,\delta_i\rangle \times \langle\delta_j+1,\beta_j\rangle \times \Pi_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\right)\} \\ & \cup\{\cup_{j=1}^{n} \left(\Pi_{i=1}^{j-1}\langle\gamma_i,\delta_i\rangle \times \langle\alpha_j,\gamma_j-1\rangle \times \Pi_{i=j+1}^{n}\langle\alpha_i,\delta_i\rangle\right)\}. \qquad (3.5) \end{aligned}$$

By lemma 3.4, the revised domain $(\langle\alpha,\beta\rangle\setminus\langle\bar{\alpha},\bar{\beta}\rangle)\setminus\langle\gamma,\delta\rangle$ is partitioned into a union of some integer subboxes. In each new generated integer subboxes, the lower bound can be calculated by solving a linear programming problem in Section 2 and the domain cut technique mentioned above can also be applied accordingly.

4 The Main Algorithm

The presented algorithm incorporates the lower bound in Section 2 and a special domain cut and partition technique in Section 3 into the branch and bound method. By the domain cut in Section 3, remove some integer subboxes that don't contain the optimal solutions to the problem (P) from the integer box $\langle l,u\rangle$, thus the hyper-rectangle region gets smaller and also the duality gap is reduced. Then the revised domain is decomposed into a union of several integer subboxes, over which the subproblems are generated. The lower bound can be calculated easily via a linear programming problem and its Lagrangian dual problem in Section 2 over each generated integer subboxes. If the lower bound of a subproblem is greater than or equal to the upper bound (the function value of the incumbent solution) of the problem (P), then we can conclude that there are no feasible solutions to this subproblem better than the incumbent. So this subproblem can be pruned in advance. If there are no feasible solutions to this problem, this subproblem can also be discarded. The domain cut and partition technique continues to be performed over the remaining integer subboxes after pruning. In this iteration process, if a feasible integer solution to (P) is found, then update the incumbent solution. Therefore, the lower bound of the problem (P) is increasing and the upper bound is decreasing. After a finite number of iterations, the optimal solution to the problem (P) can be found quickly.

The following will describe the exact algorithm for problem (P) in details.

Algorithm 4.1. (A New Algorithm for Separable Integer Programming Problems)

Step 0. (Initialization) Set $f_{opt} = +\infty$, $X^0 = \{X\}$, $k = 0$.

Step 1. Select the integer subbox $\langle\alpha^k,\beta^k\rangle$ from X^k with the minimum lower bound.

Step 2. (Bounding and partition)

(i) (Bounding) We can obtain a lower bound $v(LP)$ with the optimal continuous solution $\tilde{x}^k$ or conclude that there are no feasible solutions in the box $\langle\alpha^k,\beta^k\rangle$ by solving the linear approximation problem (LP), and we also have $v(D)$ by solving the Lagrangian dual problem (D) using procedure 2.1.

(ii) (Fathoming) Remove the integer subbox $\langle\alpha^k,\beta^k\rangle$, if one of the following conditions is satisfied:

- There are no feasible solutions in the integer box $\langle\alpha^k,\beta^k\rangle$.

- The lower bound $LB = \max\{v(LP), v(D)\}$ in the integer box $\langle \alpha^k, \beta^k \rangle$ is greater than or equal to the upper bound f_{opt} of the problem (P).

(iii) (Domain cut and partition)

(a) If $\tilde{x}^k$ is an integer solution, when $f(\tilde{x}^k) < f_{opt}$, update the incumbent $x_{opt} := \tilde{x}^k, f_{opt} := f(\tilde{x}^k)$. Then cut off the integer box $R(\tilde{x}^k)$ from $\langle \alpha^k, \beta^k \rangle$, where $R(\tilde{x}^k)$ is defined in (3.1) for a convex objective function, (3.2) for a quadratic concave objective function, and $R(\tilde{x}^k) = \{\tilde{x}^k\}$ for others.

(b) If $\tilde{x}^k$ is not an integer solution, then obtain two integer points $x^{k,1}$ and $x^{k,2}$ by rounding $\tilde{x}^k$ up or down along the gradient direction $\nabla L(\tilde{x}^k)$ and the negative gradient direction $-\nabla L(\tilde{x}^k)$ of (LP) respectively.

$x^{k,2}$ must be infeasible and the integer box $N_1(x^{k,2})$ can be cut off from $\langle \alpha^k, \beta^k \rangle$, where $N_1(x^{k,2})$ is defined in (3.3).

If $x^{k,1}$ is a feasible solution, when $f(x^{k,1}) < f_{opt}$, update the incumbent $x_{opt} := x^{k,1}, f_{opt} = f(x^{k,1})$. Cut off the integer box $R(x^{k,1})$ defined in (3.1) for the convex objective function, (3.2) for the quadratic concave objective function, and $R(x^{k,1}) = \{\tilde{x}^k\}$ for others;

If $x^{k,1}$ is infeasible, then cut off the integer box $N_2(x^{k,1})$ defined in (3.4) where x^1 is replaced by $x^{k,1}$ and α_j, β_j are replaced by $(\alpha^k)_j, (\beta^k)_j$.

After the domain cut, the remaining domain is partitioned into a union of some integer subboxes by lemma 3.4 and add the generated new subproblems to the set Y^{k+1}.

Step 3. Set $X^{k+1} = Y^{k+1} \bigcup (X^k \setminus \langle \alpha^k, \beta^k \rangle)$

Step 4. (Termination) If X^{k+1} is empty, then stop and x_{opt} is an optimal solution to (P). Otherwise, set $k := k+1$, goto Step 1.

Theorem 4.2. *The algorithm terminates at an optimal solution of (P) within a finite number of iterations.*

Proof. During the domain cut and partition process in Step2, At least $x^{k,1}$ and $x^{k,2}$ are cut off from the integer box $\langle \alpha, \beta \rangle$. Also no optimal solution can be removed . Therefore, the incumbent solution x_{opt} must be the optimal solution to (P) when the algorithm stops in Step 4 with $X^{k+1} = \emptyset$. The finite termination of the algorithm is clear due to the finiteness of X. □

5 Computational Experiments

The algorithm has been coded by Fortran 90 and has run on PC with Pentium(R) Dual-core CPU E6700@3.2GHz. There are two kinds of objective functions for the test problems in the experiment. They are of the following forms:

Type 1:

$$\begin{aligned} \min\ & f(x) = \sum_{j=1}^{n} (c_j x_j^4 + d_j x_j^3 + e_j x_j^2 + h_j x_j) \\ \text{s.t.}\ & Ax \le b, \\ & x \in X = \{x \mid l_j \le x_j \le u_j,\ x_j \text{ integer},\ j = 1, \dots, n\}, \end{aligned}$$

where c_j, d_j, and e_j are positive real numbers for the convex objective function, negative real numbers for the concave objective function and arbitrary real numbers for the nonconvex and nonconcave objective function. For each n, 10 test problems are randomly generated by a uniform distribution. For convex problems, take $c_j \in (0,1)$, $d_j \in [1,6]$, $e_j \in [1,10]$,

Table 1: Numerical results for quadratic convex problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
15×30	3.734	153.188	49.177	808	7721	3536.1	63	646	274.2
20×15	0.922	119.844	35.414	3120	224101	73467.6	251	14743	4900.2
20×20	0.547	128.859	37.220	939	167037	52861.9	47	10758	3327.4
20×30	9.750	176.641	74.492	5778	74616	37528.1	361	4169	2199.4
25×20	30.672	7477.016	1369.394	59591	5625816	1138243.2	3160	306179	57925.9
25×25	9.125	4456.922	603.753	8289	2471303	350477.3	511	116619	16606.9
25×30	70.125	2824.578	1148.253	32604	1242280	558306.7	1595	60539	27703.3
30×5	0.109	147.953	37.620	92	150250	37553.7	29	48557	7296.9

$h_j \in [-10, 10]$ and $c_j \in (-1, 0)$, $d_j \in (-1, 0)$, $e_j \in [-10, -1]$, $h_j \in [-5, 5]$ for concave problems. In addition, take $c_j = 0$ for the cubic objective function and $c_j = 0$, $d_j = 0$ for the quadratic objective function. For nonconvex and nonconcave problems, take $c_j \in [-1, 1]$, $d_j \in [-30, 30]$, $e_j \in [-40, 40]$, $h_j \in [-50, 50]$ for the quartic function; $c_j = 0$, $d_j \in [-1, 1]$, $e_j \in [-8, 8]$, $h_j \in [-50, 50]$ for the cubic function; $c_j = 0$, $d_j = 0$, $e_j \in [-1, 1]$, $h_j \in [-50, 50]$ for the quadratic function.

Type 2:

$$\begin{aligned} \min \ & f(x) = \sum_{j=1}^{n} [c_j \ln(x_j) + d_j x_j] \\ \text{s.t.} \ & Ax \le b, \\ & x \in X = \{x \mid l_j \le x_j \le u_j,\ x_j \text{ integer},\ j = 1, \ldots, n\}, \end{aligned}$$

where $c_j,\ j = 1, \ldots, n$ are negative real numbers for the convex objective function, positive real numbers for the concave objective function, and arbitrary real numbers for the nonconvex and nonconcave objective function. For each n, 10 test problems are randomly generated by a uniform distribution. Take $c_j \in (-1, 0)$ for convex problems, $c_j \in (0, 1)$ for concave problems, and $c_j \in [-1, 1]$ for nonconvex and nonconcave problems. In addition, take $d_j \in [-20, -10]$ in all test problems.

For all test problems, the constraint matrix $A = (a_{ij})_{m \times n} \in [-20, 20]$, $i = 1, \ldots m$, $j = 1, \ldots n$, $b_i = \min(\sum_{j=1}^{n} a_{ij} x_j) + r * [\max(\sum_{j=1}^{n} a_{ij} x_j) - \min(\sum_{j=1}^{n} a_{ij} x_j)]$, $i = 1, \ldots, m$, and $l_j = 1$, $u_j = 5$, $j = 1, \ldots, n$, $r = 0.6$.

In the computational experiments, take the given maximum iteration number $M = 50$ and stop criteria $\epsilon = 0.001$ in Procedure 2.1. Tables 1 - 13 summarize the numerical results, where min, max and avg stand for minimum, maximum and average respectively.

The performance of Algorithm 4.1 has been compared with the traditional branch and bound method for a quadratic convex objective function. The comparison results are reported in Table 4 where average CPU time, average subbox number (or average branches) and average iterations are obtained by running 10 test problems for each n.

From Table 4, it is clear that the proposed algorithm is much better than the traditional branch and bound method in terms of average CPU time. This main reason is the innovation of the presented algorithm which lies in diminishing the duality gap gradually by a special domain cut technique and calculating its lower bound by solving easily a linear programming problem and its Lagrangian dual problem.

Table 2: Numerical results for cubic convex problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
15×15	2.203	10.859	6.217	7117	37720	21878.3	556	3377	1884.2
15×20	2.750	10.938	6.147	5468	24713	11665.3	366	2029	989.3
15×30	5.578	118.828	52.863	1162	11924	5448.1	72	926	425.6
20×5	0.125	93.031	22.498	170	162661	39219.0	25	27105	5494.1
20×15	65.625	1947.359	681.631	100000	3862773	1580176.1	13826	280376	120580.3
20×20	82.609	681.531	318.513	95462	868343	445457.5	5439	58382	30327.4
20×30	26.203	255.828	115.847	16123	133289	68404.9	921	7883	3989.5

Table 3: Numerical results for quartic convex problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
15×15	1.891	50.625	12.838	5472	158629	39326.0	419	16380	3765.0
15×20	4.094	18.078	8.734	8531	29834	16339.9	754	2775	1403.6
15×30	17.938	89.938	55.800	3603	15059	7297.4	246	1437	591.7
20×5	0.031	160.594	23.759	19	236412	33002.2	1	45824	5449.2
20×15	56.234	3660.391	880.166	100000	7765867	1995763.0	14650	724937	165329.1
20×20	52.672	734.953	334.711	74667	922503	458018.0	3932	65804	31001.5
20×30	21.672	414.844	198.272	10994	214976	108917.3	613	13540	6462.4

Table 4: Comparison results with the traditional branch and bound method

$n \times m$	Algorithm 4.1			Traditional BB		
	Avg CPU	Avg iters	Avg boxes	Avg CPU	Avg iters	Avg branches
10×10	0.034	54.5	450.6	18.395	379.8	378.8
10×15	0.064	54.2	397.7	15.725	321.2	320.2
10×20	0.244	50.2	408.5	15.417	307.2	306.2
15×5	0.834	1998.1	16396.6	21.623	280.4	279.4
15×10	2.088	1572.4	17059.0	134.420	1740.6	1739.6
15×15	1.691	870.6	9542.5	208.164	2481.2	2480.2
20×5	52.175	64102.3	681613.6	138.453	1381.2	1380.2

Table 5: Numerical results for quadratic concave problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
30×10	0.464	0.016	1.313	3081.2	51	10590	130.7	4	500
50×10	3.738	0.031	26.359	12032.4	107	83580	314.1	3	2041
70×10	2.641	0.219	8.313	5643.7	365	17230	118.2	6	353
100×10	8.411	0.203	39.000	10170.3	173	45487	152.0	2	654
200×10	45.989	0.016	323.672	15984.1	1	108721	117.6	1	746
300×10	129.492	0.031	572.641	26570.9	1	123568	171.3	1	737
400×10	153.320	0.031	1156.734	16092.2	1	106579	67.2	1	354
30×20	9.591	0.156	38.500	24163.8	294	89991	1089.5	13	4209
40×20	188.277	1.313	717.734	353263.9	1688	1347783	12132.4	48	48100
50×20	306.359	1.922	1988.516	440319.0	2538	3284355	12233.9	110	95859
60×20	2154.697	45.641	13319.703	1494197.6	21063	9592826	33083.5	377	208559
100×20	2565.502	69.250	19386.656	1072157.3	22776	8137901	15031.4	318	112758
30×25	878.141	37.719	6384.469	1137861.2	32990	8018066	46661.7	1258	314032

Table 6: Numerical results for cubic concave problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
30×10	0.342	0.001	0.703	1570.4	29	3648	53.1	1	129
50×10	10.706	0.141	63.422	24368.4	268	142574	606.5	6	3704
70×10	20.975	0.109	95.063	30261.5	121	134732	511.9	2	2442
100×10	41.811	0.001	140.281	9697.9	1	32863	52.3	1	193
300×10	199.420	0.031	902.016	24053.5	1	101354	85.3	1	369
30×20	154.222	2.234	958.000	310130.0	2654	1902937	12698.3	72	76434
40×20	169.013	3.438	1100.781	224395.9	3443	1496426	7135.5	100	47679
50×20	467.148	2.078	4107.828	388770.7	1359	3451262	9241.2	29	81435
60×20	1024.091	72.672	2548.063	727084.6	45358	1969596	16019.5	1018	49943
30×25	643.002	10.422	2900.859	780051.0	9579	3450713	30640.1	324	148868
40×25	938.283	2.578	4348.016	825613.7	1504	3342168	26469.7	41	124756

Table 7: Numerical results for quartic concave problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
30×10	0.547	0.016	3.031	3006.5	56	17619	112.1	1	676
50×10	2.392	0.047	8.625	5498.8	77	19844	117.2	2	406
70×10	3.630	0.250	15.813	5721.1	300	25966	110.9	3	496
100×10	10.633	0.203	72.391	8249.3	136	52971	89.8	1	502
200×10	312.022	1.547	2931.875	105042.7	325	999978	767.9	1	7372
30×20	15.670	1.109	65.484	28211.2	1463	132314	1115.3	43	5462
40×20	107.197	0.516	568.469	148888.8	597	888832	5645.2	19	37418
50×20	1311.817	7.938	4938.094	1109759.9	6051	4749367	27442.4	140	117076
60×20	927.548	71.109	4511.172	627373.0	45134	3370997	13114.9	884	74384
30×25	193.605	3.969	949.047	217387.1	3172	1110700	8269.9	115	43149
40×25	1010.931	3.906	4437.031	777812.1	3455	3649881	23483.2	123	113174

Table 8: Numerical results for quadratic nonconvex and nonconcave problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
40×5	29.111	0.813	88.125	18339.2	399	53848	700.4	8	1907
20×10	0.109	0.016	0.234	944.5	141	2011	55.3	7	106
30×10	2.592	0.203	10.828	13926.0	879	55327	530.9	40	1973
40×10	24.689	0.969	119.250	106834.8	3860	561672	3723.5	112	21146
60×10	447.147	49.828	1630.766	886573.2	89299	2930382	16887.4	1545	46703
65×10	3487.561	69.563	13170.406	5733711.6	100000	21564731	105265.4	2327	380080
20×15	0.542	0.094	2.766	2866.2	257	15313	179.9	12	1032
30×15	11.648	0.828	45.359	36686.1	2107	167360	1382.4	71	6436
40×15	51.964	2.078	176.219	109257.2	4129	384329	3403.6	130	11423
20×20	3.983	0.156	19.719	12707.0	386	58347	805.4	17	3530
30×20	101.328	2.484	236.953	190359.0	4002	447198	8146.8	162	22416
40×20	663.378	18.344	2459.656	794163.0	23359	2614320	25805.1	633	82647
30×25	217.730	10.797	1426.766	247523.5	14498	1515520	9615.5	703	56809
40×25	655.094	37.422	2773.156	526475.1	29878	2203335	17344.5	966	76960
30×30	943.655	22.328	3985.719	701886.4	19715	2968514	27604.9	1048	121019
40×30	4810.697	131.953	32754.391	2778449.2	56846	19726675	82912.6	1799	590758

Table 9: Numerical results for cubic nonconvex and nonconcave problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
20×5	2.477	0.313	10.906	3543.1	532	16095	224.2	23	884
20×10	5.145	0.641	17.953	47445.8	6687	153005	2854.3	333	9791
25×10	20.478	3.625	47.406	143116.6	26590	305885	6100.6	994	14703
30×10	1434.822	4.563	3717.844	7324516.0	18520	19895136	278892.7	534	857136
20×15	18.730	4.797	75.703	108196.3	17497	494254	6141.4	1055	29750
25×15	609.128	22.188	2263.938	2378019.4	72308	8606078	107888.8	2631	408620
20×20	61.178	1.500	162.281	181471.8	6048	547034	10523.6	393	33637
25×20	479.922	19.609	1708.766	1304565.6	64345	4879657	64986.8	2891	280195

Table 10: Numerical results for quartic nonconvex and nonconcave problems of Type 1

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
20×5	37.989	0.359	177.328	57667.1	455	281356	3241.0	20	16592
20×10	17.402	0.672	35.641	153545.6	6782	324439	8256.2	294	17610
25×10	1231.808	3.484	6893.625	6359249.5	18359	31816011	297796.0	578	1506535
20×15	141.058	7.063	844.266	695470.0	33932	4078943	44737.4	1606	290255
20×20	224.664	10.609	1169.281	784369.9	23364	4506699	54384.6	983	342667

Table 11: Numerical results for convex problems of Type 2

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
20×10	0.417	0.109	1.078	366.1	86	1017	27.0	5	75
40×10	5.416	0.766	13.703	1842.0	121	4819	71.6	6	167
60×10	16.864	1.984	62.141	2945.5	350	11315	75.6	5	295
80×10	143.453	2.563	626.219	16396.7	206	72964	302.8	2	1318
100×10	843.055	16.500	6519.969	72200.5	1260	553271	1163.9	21	8646
20×15	4.016	0.141	10.047	3381.3	97	9921	217.1	6	677
40×15	113.188	1.563	933.172	32980.8	302	270857	1094.9	11	9023
60×15	644.122	4.016	1899.141	101942.0	442	306291	2849.6	9	8622

Table 12: Numerical results for concave problems of Type 2

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
20×10	0.461	0.078	1.500	496.0	89	1533	28.0	4	83
40×10	6.041	0.438	16.078	2562.1	158	6884	79.1	6	205
60×10	18.109	2.078	74.234	4127.2	348	17700	82.5	5	355
80×10	121.969	1.953	517.578	16910.5	206	74302	258.0	2	1166
100×10	1005.416	15.359	7180.078	114485.9	1662	801438	1611.3	20	11699
20×15	3.977	0.188	9.813	3738.6	128	9368	219.7	6	612
40×15	138.458	2.047	1182.500	50142.1	546	427089	1634.0	11	14148
60×15	683.477	3.484	2568.406	143582.6	531	550901	3619.7	9	14076

Table 13: Numerical results for nonconvex and nonconcave problems of Type 2

$n \times m$	CPU Time (seconds)			Number of Subboxes			Number of Iterations		
	avg	min	max	avg	min	max	avg	min	max
20×10	0.517	0.125	1.625	558.2	124	1860	31.8	6	96
40×10	9.034	0.969	24.953	3821.0	374	10860	116.4	10	330
60×10	33.284	3.984	100.875	7588.8	790	23890	149.7	12	472
80×10	296.617	5.531	1521.625	43526.7	657	225560	663.0	7	3551
100×10	3042.853	30.391	24811.516	348118.9	3298	2817407	4853.7	53	40100
20×15	4.355	0.203	12.156	4032.8	129	10707	235.6	6	614
40×15	161.566	3.641	1353.328	58302.9	1020	487810	1900.9	22	16223
60×15	1006.855	4.938	3938.547	207552.4	794	837453	4980.5	13	20885

For the problems of Type 1, from Tables 1 - 10 we can find that it usually spends much more time with the degree increasing for the problems with the same dimension, since the problems become more complicated when the degree increases. For different dimension problems with the same degree objective function, the CPU time for the algorithm usually increases with the dimensions increasing. Sometimes the abnormal situation may arise mainly due to the random generated data. In Table 10, we see that it spends 1231.808 seconds when solving the problem with $n = 25$ and $m = 10$. This case occurs, on the one hand, owing to the random generated data. On the other hand, it shows that the nonconvex and nonconcave problem is more difficult to solve when the degree increases. This is because more integer subboxes are generated in order to ensure that $f_j(x)$ is a either convex or concave function over the subintervals when we calculate the lower bound via the linear underestimation problem for the nonconvex and nonconcave problems in Section 2 and more subproblems will be solved.

For the problems of Type 2, we can also observe that solving the nonconvex and nonconcave problems takes much more time than solving the convex and concave problems with the same dimension, since the nonconvex and nonconcave problems are more complicated than the convex and concave problems with the same dimension.

From the above results in Tables 1 - 13, we can observe that the algorithm can find the exact solutions of medium-scale separable integer programming problems in reasonable computation time. According to our computational numerical results, the CPU time spent by the algorithm mainly depends both on the number of subboxes and on the computation time to solve the dual problem using the subgradient method and search for the optimal solution to the linear approximation problem in each subbox. If the lower bound is not very good, then more integer subboxes will be left for further consideration. Thus much more time is needed for solving these subproblems. Therefore, if the lower bound can be further improved, the algorithm will be more efficient.

6 Concluding Remarks

A new exact algorithm for nonlinear separable integer programming problems is proposed in this paper. The algorithm incorporates a new cut strategy into the branch and bound method. The domain cut technique makes the algorithm different from the traditional branch and bound method. In the domain cut and partition process, removing the domains that don't contain the optimal solution of the primal problem makes the feasible region shrink greatly. The lower bound is taken as the maximum of the optimal value of the linear approximation problem and the Lagrangian dual value, which ensures that we can get a better lower bound to fathom more integer subboxes. The lower bound of the primal problem is increasing gradually and the duality gap is decreasing. Thus the optimal solution of the primal problem can be found quickly in a finite numbers of iterations. The efficiency of the proposed algorithm can be witnessed from the above computation experiments in Tables 1 - 13. Finally, the algorithm presented in this paper can also be extended to solve general separable integer programming problems with nonlinear constraints.

References

[1] A. Beck and M. Teboulle, Global optimality conditions for quadratic optimization problems with binary constraints, *SIAM J. Optim.* 11 (2000) 179–188.

[2] R. Bellman and S.E. Dreyfus, *Applied Dynamic Programming*, Princeton, Princeton University Press, 1962.

[3] H.P. Benson and SS. Erenguc, An algorithm for concave integer minimization over a polyhedron, *Naval Res. Logist.* 37 (1990) 515–525.

[4] H. P. Benson, S.S. Erengue and R. Horst, A note on adopting methods for continuous global optimization to the discrete case, *Ann. Oper. Res.* 25 (1990) 243–252.

[5] K.M. Bretthauer, A.V. Cabot and M.A. Venkataramanan, An algorithm and new penalties for concave integer minimization over a polyhedron, *Naval Res. Logist.* 41 (1994) 435–454.

[6] K.M. Bretthauer and B. Shetty, The nonlinear resource allocation problem, *Oper. Res.* 43 (1995) 670–683.

[7] K.M. Bretthauer and B. Shetty, The nonlinear knapsack problem–algorithms and applications, *Eur. J. Oper. Res.* 138 (2002a) 459–472.

[8] K.M. Bretthauer and B. Shetty, A pegging algorithm for the nonlinear resource allocation p roblem, *Comput. Oper. Res.* 29 (2002b) 505–527.

[9] A.V. Cabot and S.S. Erengue, A Branch and Bound Algorithm for Solving a Class of Nonlinear Integer Programming Problems, *Naval Res. Logist.* 33 (1986) 559–567.

[10] M. Held and R.M. Karp, A dynamic programming approach to sequencing problems, *J. Soc. Ind. Appl. Math.* 10 (1962) 196–210.

[11] D. Hochbaum, A nonlinear knapsack problem, *Oper. Res. Lett.* 17 (1995) 103–110.

[12] R. Horst, P.M. Pardalos and N.V. Thoai, *Introduction to Global Optimization, Second Edition, Nonconvex optimization and its application*, Kluwer, Dordrecht, Netherlands, 2000.

[13] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, Springer-Verlag, Heidelberg, 1993.

[14] R.A. Howard, Dynamic programming, *Manage. Sci.* 12 (1966) 317–348.

[15] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*, Cambridge, Mass., MIT Press, 1988.

[16] M.S. Kodialam and H. Luss, Algorithm for separable nonlinear resource allocation problems, *Oper. Res.* 46 (1998) 272–284.

[17] J. B. Lasserre, An explicit equivalent positive semidefinite program for nonlinear 0-1 programs, *SIAM J. Optim.* 12 (2002) 756–769.

[18] D. Li, X.L. Sun and F.L. Wang, Convergent Lagrangian and contour cut method for nonlinear integer programming with a quadratic objective function, *SIAM J. Optim.* 17 (2006) 372–400.

[19] D. Li, X.L. Sun, J. Wang and K.I.M. McKinnon, Convergent Lagrangian and domain cut method for nonlinear knapsack problem, *Comput. Optim. Appl.* 42 (2009) 67–104.

[20] R.E. Marsten and T.L. Morin, A hybrid approach to discrete mathematical programming, *Math. Program.* 14 (1978) 21–40.

[21] K. Mathur, H.M. Salkin and S. Morito, A branch and search algorithm for a class of nonlinear knapsack problems, *Oper. Res. Lett.* 2 (1983) 55–60.

[22] T.L. Morin and R.E. Marsten, An algorithm for nonlinear knapsack problems, *Manage. Sci.* 22 (1976a) 1147–1158.

[23] T.L. Morin and R.E. Marsten, Branch and bound strategies for dynamic programming, *Oper. Res.* 24 (1976b) 611–627.

[24] P.M. Pardalos and J.B. Rosen, Reduction of nonlinear integer separable programming problems, *Int. J. Comput. Math.* 24 (1988) 55–64.

[25] X.L. Sun and D. Li, Optimality condition and branch and bound algorithm for constrained redundancy optimization in series systems, *Optim. Eng.* 3 (2002) 53–65.

[26] F.L. Wang, A new exact algorithm for concave knapsack problems with integer variables, *Int. J. Comput. Math.* 96 (2019) 126-134.

Manuscript received 8 March 2019
revised 24 May 2019
accepted for publication 21 July 2019

Fenlan Wang
College of Science
Nanjing University of Aeronautics and Astronautics
29 Yudao St., Nanjing 210016, P. R. China
E-mail address: flwang@nuaa.edu.cn