Journal of Global Optimization (2005) 33: 15–30 © Springer 2005
DOI 10.1007/s10898-005-1678-6

Exact Algorithm for Concave Knapsack Problems: Linear Underestimation and Partition Method*

X. L. SUN[1], F. L. WANG[1] and D. LI[2]
[1]*Department of Mathematics, Shanghai University, 99 Shangda Road, Baoshan, Shanghai 200444, P. R. China*
[2]*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong* (*e-mail: dli@se.cuhk.edu.hk*)

(Received 21 June 2004; accepted: 30 January 2005)

Abstract. Integer programming problems with a concave cost function are often encountered in optimization models involving economics of scale. In this paper, we propose an efficient exact algorithm for solving concave knapsack problems. The algorithm consists of an iterative process between finding lower and upper bounds by linearly underestimating the objective function and performing domain cut and partition by exploring the special structure of the problem. The lower bound is improved iteratively via cutting and partitioning the domain. This iteration process converges to the optimality in a finite number of steps. Promising computational results are reported for large-scale concave knapsack problems with up to 1200 integer variables. Comparison results with other existing methods in the literature are also presented.

Key words: concave knapsack problem, domain cut, domain partition, linear underestimation, nonlinear integer programming.

1. Introduction

Consider the following concave knapsack problem:

$$\begin{aligned} (P) \quad \min \quad & f(x)=\sum_{j=1}^{n} f_j(x_j), && (1)\\ \text{s.t.} \quad & g(x)=\sum_{j=1}^{n} b_j x_j \leqslant b, \\ & x\in X=\{x \mid l_j \leqslant x_j \leqslant u_j,\ x_j \text{ integer},\ j=1,\dots,n\}, \end{aligned}$$

where f_j, $j=1,\dots,n$, are nonincreasing concave functions on $\mathbb{R}$, $b_j>0$, $j=1,\dots,n$, and l_j and u_j are integer lower and upper bounds of x_j. Set X is termed the domain of the decision variables.

*Research supported by the National Natural Science Foundation of China under Grants 79970107 and 10271073, and the Research Grants Council of Hong Kong under Grant CUHK 4214/01E.

Nonlinear separable integer programming has received considerable attention during the last 30 years. Most of the research efforts have been devoted to convex separable integer programming problems where f_j's are convex functions. Branch-and-bound methods and their combination with dynamic programming were proposed for nonlinear resource allocation problems [5–7, 15, 25] and convex nonlinear knapsack problems [11, 18, 19]. Hochbaum [12] proposed a 0–1 linearization method to convert a convex nonlinear knapsack problem into an equivalent 0–1 linear integer program. Pardalos and Rosen [23] proposed a linearizarion method to reduce a general separable integer programming problem into a linear 0–1 problem. Marsten and Morin [16], Morin and Marsten [20, 21] presented dynamic programming methods for nonlinear separable integer programming problems (see [9] and the references therein for the use of dynamic programming methods in nonlinear integer programming). Solution methods for nonlinear integer knapsack problems with a special form of constraint: $\sum_{j=1}^{n} x_j = b$ are summarized in Ibaraki and Katoh's book [14]. An $O(n)$ algorithm was proposed by Pardalos and Kovoor [22] for the continuous optimization problem of minimizing a convex quadratic function subject to $\sum_{j=1}^{n} x_j = b$ and $0 \leqslant x_j \leqslant \beta_j$, $j = 1, \ldots, n$.

In real-world applications, however, nonconvex objective functions are often encountered in nonlinear integer programming models. The concave cost function is of particular interests due to its applications in problems involving economics of scale [24], capacity planning and fixed-charge problems with integer variables [5, 13]. The hybrid dynamic programming method developed in [16] can deal with nonconvex integer programming problems. The method generates efficient feasible solutions recursively and uses lower bounds in the course of the recursion to fathom efficient solutions that are unable to lead to an optimal solution. The method needs to store all the unfathomed efficient solutions during the algorithm and will cause storage problem as the dimension n increases. Moveover, in [16], the lower bound is obtained via converting a residual subproblem into a 0–1 linear programming and then solving its continuous relaxation by dual simplex methods. In nonlinear cases, this solution method introduces many extra variables. Numerical results were reported in [16] only for 0–1 linear integer programming problems. A conventional branch-and-bound method combined with a linear underestimation strategy was suggested in [5] for solving concave resource allocation problems, while no computational results were reported. Branch-and-bound algorithms based on the global optimization methods for concave minimization were proposed in [2–4, 8] for solving concave integer programming problems over a polyhedron. Since solution methods for concave minimization are based on outer approximation and vertex ranking of a polyhedron, branch-and-bound methods using direct continuous relaxation can be only applicable to solve small size problems.

This paper aims to develop a new exact method for problem (P). The basic idea underlying the proposed algorithm is adopting lower bounding by linear approximation and revising the domain of the decision variables by cutting and partitioning. Approximating the concave function f_j's by its convex underestimation leads to a linear knapsack subproblem. The continuous optimal solution of the linear knapsack subproblem can be obtained by a simple greedy procedure. It is shown that the resulting lower bound coincides with the optimal Lagrangian dual value of the subproblem. To improve the lower bound, certain integer points are cut from the domain. Due to the special property of the problem, it is proved that an integer box after a cut can be partitioned into a union of at most $n-1$ smaller integer boxes. For each integer box included in a revised domain, a lower bound and an upper bound are obtained. All the integer boxes with the lower bound greater than or equal to the incumbent will be fathomed. The algorithm generates a sequence of strictly increasing lower bounds of the problem (P) and terminates when there is no active integer box. Computational results show that the algorithm is capable of solving large-scale concave knapsack problems with up to 1200 integer variables in reasonable computation time. The efficiency of the algorithm is also confirmed by comparing it with other existing methods in the literature.

2. Linear Approximation and Bounding

Let $\alpha,\ \beta \in \mathbb{Z}^n$, where $\mathbb{Z}^n$ denotes the set of integer points in $\mathbb{R}^n$. Denote by $[\alpha,\beta]$ the box (hyper-rectangle) formed by α and β, $[\alpha,\beta]=\{x \mid \alpha_j \leqslant x_j \leqslant \beta_j,\ j=1,\dots,n\}$. Denote by $\langle\alpha,\beta\rangle$ the set of integer points in $[\alpha,\beta]$,

$$\langle\alpha,\beta\rangle=\{x \mid \alpha_j \leqslant x_j \leqslant \beta_j,\ x_j \text{ integer},\ j=1,\dots,n\}=\prod_{j=1}^{n}\langle\alpha_j,\beta_j\rangle.$$

The set $\langle\alpha,\beta\rangle$ is called an integer box. For convenience, we define $[\alpha,\beta]=\langle\alpha,\beta\rangle=\emptyset$ if $\alpha \nleqslant \beta$

Let $\langle\alpha,\beta\rangle \subseteq X=\{x \mid l_j \leqslant x_j \leqslant u_j,\ x_j \text{ integer},\ j=1,\dots,n\}$ be a nonempty integer box. Consider the following subproblem of (P):

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

Denote by $v(\cdot)$ the optimal value of problem $(\cdot)$. The convex underestimating function of $f(x)=\sum_{j=1}^{n} f_j(x_j)$ over box $[\alpha,\beta]$ can be expressed as:

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

where $L_j(x_j)=f_j(\alpha_j)-a_j(x_j-\alpha_j)$ with

$$a_j=\begin{cases} -\frac{f_j(\beta_j)-f_j(\alpha_j)}{\beta_j-\alpha_j}, & \alpha_j<\beta_j,\\ 0, & \alpha_j=\beta_j.\end{cases}$$

The linear approximation of (SP) is

$$\begin{aligned} \text{(LSP)}\quad \min\quad & L(x)=a_0-\sum_{j=1}^{n} a_j x_j,\\ \text{s.t.}\quad & g(x)=\sum_{j=1}^{n} b_j x_j \leqslant b, \quad x\in\langle\alpha,\beta\rangle, \end{aligned}$$

where $a_0=\sum_{j=1}^{n}[f_j(\alpha_j)+a_j\alpha_j]$ is the constant term of $L(x)$. Since f_j, $j=1,\dots,n$, are nonincreasing functions, we have $a_j\geqslant 0$ for $j=1,\dots,n$. Without loss of generality, we assume that

$$\frac{a_1}{b_1}\geqslant\frac{a_2}{b_2}\geqslant\dots\geqslant\frac{a_n}{b_n}.$$

Let

$$\xi_j=\left(b-\sum_{i=1}^{j-1} b_i\beta_i-\sum_{i=j+1}^{n} b_i\alpha_i\right)/b_j,\quad j=1,\dots,n. \tag{2}$$

Let k be the largest index j satisfying $\xi_j>\alpha_j$. It is well known from [10] that the optimal solution of the continuous relaxation of (LSP) is

$$x^R=(\beta_1,\dots,\beta_{k-1},\xi_k,\alpha_{k+1},\dots,\alpha_n)^T. \tag{3}$$

Let $\tau_k=\lfloor\xi_k\rfloor$, where $\lfloor\xi_k\rfloor$ denote the maximum integer equal to or less than ξ_k. A feasible solution can be derived from x^R by replacing ξ_k with τ_k

$$x^F=(\beta_1,\dots,\beta_{k-1},\tau_k,\alpha_{k+1},\dots,\alpha_n)^T. \tag{4}$$

From (2) and (4), we infer that if $\xi_k = \tau_k$, then $x^F = x^R$ is an optimal solution to (LSP). Suppose that $\xi_k \neq \tau_k$. Let

$$x^I = (\beta_1, \beta_2, \dots, \beta_{k-1}, \tau_k + 1, \alpha_{k+1}, \dots, \alpha_n)^T. \tag{5}$$

It follows that $x^I \in \langle \alpha, \beta \rangle$ and x^I is infeasible. Let (RLSP) denote the continuous relaxation problem of (LSP). Then, from the above discussion, we have

$$L(x^R) = v(RLSP) \leqslant v(LSP) \leqslant v(SP) \leqslant f(x^F).$$

Therefore, by solving $(RLSP)$, we can get a lower bound $L(x^R)$ and an upper bound $f(x^F)$ of the subproblem (SP).

It is interesting to compare $L(x^R)$ with the lower bound provided by Lagrangian dual problem of (SP). The Lagrangian dual problem of (SP) is

$$\text{(SD)} \quad \max_{\lambda \geqslant 0} d(\lambda),$$

where $d(\lambda)$ is the dual function defined by

$$d(\lambda) = \min_{x \in \langle \alpha, \beta \rangle} l(x, \lambda) = f(x) + \lambda \left(\sum_{j=1}^{n} b_j x_j - b \right). \tag{6}$$

The following theorem shows that $L(x^R)$ coincides with the optimal Lagrangian dual value of problem (SP).

THEOREM 1. $v(SD) = v(RLSP) = L(x^R)$.

Proof. Since $f(x)$ is concave, the Lagrangian function $l(x, \lambda)$ in (6) is a concave function of x for any $\lambda \geqslant 0$. Thus, it always achieves its minimum over $[\alpha, \beta]$ at one of its extreme points, which are in $\langle \alpha, \beta \rangle$. On the other hand, $f(x)$ takes the same values as $L(x)$ over all the extreme points of box $[\alpha, \beta]$. Therefore, we have

$$\begin{aligned}
v(SD) &= \max_{\lambda \geqslant 0} d(\lambda) \\
&= \max_{\lambda \geqslant 0} \min_{x \in \langle \alpha, \beta \rangle} f(x) + \lambda \left(\sum_{j=1}^{n} b_j x_j - b \right) \\
&= \max_{\lambda \geqslant 0} \min_{x \in [\alpha, \beta]} L(x) + \lambda \left(\sum_{j=1}^{n} b_j x_j - b \right) \\
&= \min_{x \in [\alpha, \beta]} \max_{\lambda \geqslant 0} L(x) + \lambda \left(\sum_{j=1}^{n} b_j x_j - b \right)
\end{aligned}$$

$$= \min \left\{ L(x) \mid \sum_{j=1}^{n} b_j x_j \leqslant b,\ \ x \in [\alpha, \beta] \right\} = v(RLSP) = L(x^R).$$

The fourth equality is due to the duality theorem of linear programming.

3. Partition and the Solution Method

The key idea underlying the proposed method is to remove from X certain integer boxes that for sure do not include optimal solution of (P) and then partition the revised domain into a union of integer boxes. Let $A = \langle \alpha, \beta \rangle$, $B^F = \langle \alpha, x^F \rangle$ and $B^I = \langle x^I, \beta \rangle$. By the monotonicity of $f(x)$, cutting integer box B^F from A does not remove any feasible solution better than x^F. Moreover, cutting integer box B^I does not remove any feasible solution from A. Let $\Omega = (A \backslash B^F) \backslash B^I$. We will show in the following that Ω can be partitioned into a union of at most $n-1$ integer boxes. A lower bound and an upper bound on Ω can be then calculated by applying the linear approximation approach proposed in the previous section to each integer box.

LEMMA 1. Let $A = \langle \alpha, \beta \rangle$ and $B = \langle \gamma, \delta \rangle$, where α, β, γ, $\delta \in \mathbb{Z}^n$ and $\alpha \leqslant \gamma \leqslant \delta \leqslant \beta$. Then

$$A \backslash B = \left\{ \bigcup_{j=1}^{n} \left(\prod_{i=1}^{j-1} \langle \alpha_i, \delta_i \rangle \times \langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle \right) \right\}$$
$$\bigcup \left\{ \bigcup_{j=1}^{n} \left(\prod_{i=1}^{j-1} \langle \gamma_i, \delta_i \rangle \times \langle \alpha_j, \gamma_j - 1 \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \delta_i \rangle \right) \right\}. \tag{7}$$

Proof. As illustrated in Figure 1, $A \backslash B$ can be expressed as

$$A \backslash B = \langle \alpha, \beta \rangle \backslash \langle \gamma, \delta \rangle = (\langle \alpha, \beta \rangle \backslash \langle \alpha, \delta \rangle) \bigcup (\langle \alpha, \delta \rangle \backslash \langle \gamma, \delta \rangle). \tag{8}$$

Let $C = \langle \alpha, \delta \rangle$. Then, by (8),

$$A \backslash B = (A \backslash C) \bigcup (C \backslash B). \tag{9}$$

For $j = 0, 1, \ldots, n-1$, define

$$A_j = \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle, \quad C_j = \prod_{i=j+1}^{n} \langle \alpha_i, \delta_i \rangle.$$

Then

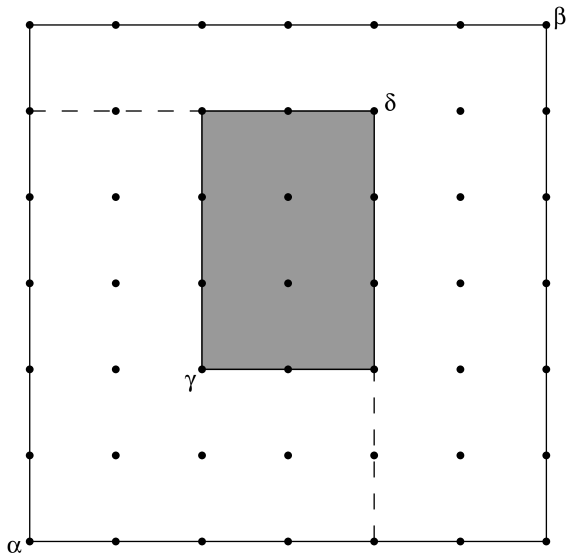

Figure 1. Partition of set $A\backslash B$.

$$\begin{aligned}
&A_{j-1}\backslash C_{j-1}\\
&=\prod_{i=j}^{n}\langle\alpha_i,\beta_i\rangle\backslash\prod_{i=j}^{n}\langle\alpha_i,\delta_i\rangle\\
&=\left\{(\langle\alpha_j,\delta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle)\bigcup(\langle\delta_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle)\right\}\backslash\prod_{i=j}^{n}\langle\alpha_i,\delta_i\rangle\\
&=\left\{(\langle\alpha_j,\delta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle)\backslash\prod_{i=j}^{n}\langle\alpha_i,\delta_i\rangle\right\}\bigcup(\langle\delta_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle)\\
&=\left\{\langle\alpha_j,\delta_j\rangle\times(\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\backslash\prod_{i=j+1}^{n}\langle\alpha_i,\delta_i\rangle)\right\}\bigcup(\langle\delta_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle)\\
&=\{\langle\alpha_j,\delta_j\rangle\times(A_j\backslash C_j)\}\bigcup\left(\langle\delta_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\right). \qquad (10)
\end{aligned}$$

Notice that $A=A_0$, $C=C_0$, $A_{n-1}\backslash C_{n-1}=\langle\alpha_n,\beta_n\rangle\backslash\langle\alpha_n,\delta_n\rangle=\langle\delta_n+1,\beta_n\rangle$. Using the partition formulation (10) recursively for $j=1,\dots,n-1$, we get

$$A\backslash C=\bigcup_{j=1}^{n}\left(\prod_{i=1}^{j-1}\langle\alpha_i,\delta_i\rangle\times\langle\delta_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\right). \qquad (11)$$

Similarly, we have

$$C\backslash B=\bigcup_{j=1}^{n}\left(\prod_{i=1}^{j-1}\langle\gamma_i,\delta_i\rangle\times\langle\alpha_j,\gamma_j-1\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\delta_i\rangle\right). \tag{12}$$

Combining (9) with (11) and (12) yields (7). □

Now, consider the partition of set $\Omega=(A\backslash B^F)\backslash B^I$, where $B^F=\langle\alpha,x^F\rangle$ and $B^I=\langle x^I,\beta\rangle$ with x^F and x^I defined by (4) and (5), respectively. Since x^F and x^I are on the boundary of the box $A=\langle\alpha,\beta\rangle$, we are able to partition Ω into at most $n-1$ subboxes as stated in the following corollary.

COROLLARY 1. The set $\Omega=(A\backslash B^F)\backslash B^I$ can be partitioned into at most $n-1$ integer boxes

$$\begin{aligned}\Omega=&\left\{\bigcup_{j=1}^{k-1}\left(\prod_{i=1}^{j-1}\langle\beta_i,\beta_i\rangle\times\langle\alpha_j,\beta_j-1\rangle\times\prod_{i=j+1}^{k-1}\langle\alpha_i,\beta_i\rangle\right.\right.\\&\left.\left.\times\langle\tau_k+1,\beta_k\rangle\times\prod_{i=k+1}^{n}\langle\alpha_i,\beta_i\rangle\right)\right\}\bigcup\left\{\bigcup_{j=k+1}^{n}\left(\prod_{i=1}^{k-1}\langle\alpha_i,\beta_i\rangle\times\langle\alpha_k,\tau_k\rangle\right.\right.\\&\left.\left.\times\prod_{i=k+1}^{j-1}\langle\alpha_i,\alpha_i\rangle\times\langle\alpha_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\right)\right\}.\end{aligned} \tag{13}$$

Proof. By (7), we have

$$\begin{aligned}A\backslash B^F=&\left\{\prod_{i=1}^{k-1}\langle\alpha_i,\beta_i\rangle\times\langle\tau_k+1,\beta_k\rangle\right.\\&\left.\times\prod_{i=k+1}^{n}\langle\alpha_i,\beta_i\rangle\right\}\bigcup\left\{\bigcup_{j=k+1}^{n}\left(\prod_{i=1}^{k-1}\langle\alpha_i,\beta_i\rangle\times\langle\alpha_k,\tau_k\rangle\right.\right.\\&\left.\left.\times\prod_{i=k+1}^{j-1}\langle\alpha_i,\alpha_i\rangle\times\langle\alpha_j+1,\beta_j\rangle\times\prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\right)\right\}.\end{aligned}$$

It is easy to see that only the first integer box of the above union of $n-k+1$ integer boxes contains x^I. Thus,

$$\begin{aligned}(A\backslash B^F)\backslash B^I = & \left\{\prod_{i=1}^{k-1}\langle\alpha_i,\beta_i\rangle \times \langle\tau_k+1,\beta_k\rangle \right. \\ & \left. \times \prod_{i=k+1}^{n}\langle\alpha_i,\beta_i\rangle\right\} \Big\backslash \left\{\prod_{i=1}^{k-1}\langle\beta_i,\beta_i\rangle \times \langle\tau_k+1,\beta_k\rangle \right. \\ & \left. \times \prod_{i=k+1}^{n}\langle\alpha_i,\beta_i\rangle\right\} \bigcup \left\{ \bigcup_{j=k+1}^{n}\left(\prod_{i=1}^{k-1}\langle\alpha_i,\beta_i\rangle \times \langle\alpha_k,\tau_k\rangle \right.\right. \\ & \left.\left. \times \prod_{i=k+1}^{j-1}\langle\alpha_i,\alpha_i\rangle \times \langle\alpha_j+1,\beta_j\rangle \times \prod_{i=j+1}^{n}\langle\alpha_i,\beta_i\rangle\right)\right\}.\end{aligned}$$

Using (7) again, we obtain (13). □

We now describe the algorithm.

ALGORITHM 1.

Step 0 (Initialization). Let $l=(l_1,\dots,l_n)^T$, $u=(u_1,\dots,u_n)^T$. If l is infeasible then problem (P) has no feasible solution, stop. Otherwise, set $x_{best}=l$, $f_{best}=f(x_{best})$, $X^1=\langle l,u\rangle$, $Y^1=X^1$, $Z^k=\emptyset$. Set $k=1$.

Step 1 (Linear approximation). For each $\langle\alpha,\beta\rangle\in Y^k$, do the following:

(i) If $g(\alpha)>b$, then remove $\langle\alpha,\beta\rangle$ from Y^k.

(ii) Compute the linear approximation function $L(x)$. For convenience, suppose $\{a_j/b_j\}_{j=1}^n$ is already in decreasing order. Calculate the continuous optimal solution x^R by (3).

(a) If $L(x^R)\geqslant f_{\rm best}$, then remove $\langle\alpha,\beta\rangle$ from Y^k and repeat Step 1;

(b) If $\xi_k=\tau_k$, then $x^F=x^R$ is an optimal solution to the corresponding subproblem (LSP). If $f(x^F)<f_{\rm best}$ set $f_{\rm best}=f(x^F)$ and $x_{\rm best}=x^F$, remove $\langle\alpha,\beta\rangle$ from Y^k. Otherwise, goto (iii).

(iii) Calculate x^F and x^I by (4) and (5), respectively. Determine τ_j for $j=k+1,\dots,n$ by

$$\tau_j=\min\left\{\beta_j,\left\lfloor\left(b-\sum_{i=1}^{k-1}b_i\beta_i-\sum_{i=k}^{j-1}b_i\tau_i-\sum_{i=j+1}^{n}b_i\alpha_i\right)/b_j\right\rfloor\right\}. \tag{14}$$

Let

$$\bar{x}^F=(\beta_1,\beta_2,\dots,\beta_{k-1},\tau_k,\tau_{k+1},\dots,\tau_n)^T. \tag{15}$$

If $f(\bar{x}^F)<f_{\rm best}$ set $f_{\rm best}=f(\bar{x}^F)$ and $x_{\rm best}=\bar{x}^F$.

Step 2 (Fathoming). For each $\langle\alpha,\beta\rangle$, denote by $r(\alpha,\beta)$ the lower bound $L(x^R)$, the optimal value of (RLSP) on $\langle\alpha,\beta\rangle$. Let $T^k = Y^k \bigcup Z^k$. For each $\langle\alpha,\beta\rangle \in T^k$, remove $\langle\alpha,\beta\rangle$ from T^k if $r(\alpha,\beta) \geqslant f_{\text{best}}$.

Step 3 (Partition). If $T^k = \emptyset$, stop, x_{best} is an optimal solution to (P). Otherwise, find the integer box $\langle\alpha,\beta\rangle$ with minimum value of $r(\alpha,\beta)$:

$$f_k = r(\alpha,\beta) = \min_{\langle\tilde{\alpha},\tilde{\beta}\rangle \in T^k} r(\tilde{\alpha},\tilde{\beta}).$$

Set $Z^{k+1} = T^k \backslash \{\langle\alpha,\beta\rangle\}$ and

$$Y^{k+1} = (\langle\alpha,\beta\rangle \backslash \langle\alpha, x^F\rangle) \backslash \langle x^I, \beta\rangle,$$

where x^F and x^I were calculated in Step 1 (iii). Partition Y^{k+1} into a union of integer boxes by using the formula (13). Set $X^{k+1} = Y^{k+1} \bigcup Z^{k+1}$, $k := k+1$, goto Step 1.

A few remarks about the algorithm are as follows.

Remark 1. In Step 1 (ii), the necessary ranking of $\{a_j/b_j\}_{j=1}^n$ and the corresponding variable re-ordering of x can be easily done in general cases when $\{a_j/b_j\}_{j=1}^n$ is not in decreasing order.

Remark 2. Calculating $\bar{x}^F$ in Step 1 (iii) is to improve the feasible solution x^F by filling the slack of constraint at x^F. Since x^F is feasible, it follows that $\bar{x}^F$ is also feasible and $f(\bar{x}^F) \leqslant f(x^F)$. Similar heuristics have been used in generating feasible solutions for linear knapsack problems (see [17]).

Remark 3. In the algorithm, $X^k = Y^k \bigcup Z^k$ represents all the active integer boxes, where Y^k is the set of newly generated integer boxes on each of which a lower bound and an upper bound have been calculated in Step 1, and Z^k is the set of old integer boxes inherited from X^{k-1}. Each integer box in X^k is associated with a lower bound $L(x^R)$, a feasible solution x^F and an infeasible solution x^I. The incumbent x_{best} and the corresponding best function value f_{best} are obtained by comparing the last incumbent with the minimum of upper bounds, x^F or $\bar{x}^F$, on the integer boxes in Y^k.

THEOREM 2. *The algorithm generates a nondecreasing sequence of lower bounds* $\{f_k\}$ *and terminates at an optimal solution of* (P) *within a finite number of iterations.*

Proof. For each integer box $\langle\alpha,\beta\rangle$ of $X^{k+1}=Y^{k+1}\bigcup Z^{k+1}$, it is either identical to an integer box in X^k or a subset of an integer box in X^k. Thus, the linear underestimation of $f(x)$ on $\langle\alpha,\beta\rangle$ majorizes that on the corresponding integer box of X^k. Moreover, from Step 3, the continuous optimal solution x^R corresponding to the minimum lower bound f_k is excluded in X^{k+1}. Therefore, $f_{k+1}\geqslant f_k$ for $k\geqslant 1$. The finite termination of the algorithm is obvious from the finiteness of X and the fact that at least the feasible solution x^F and infeasible solution x^I corresponding to the minimum lower bound f_k are cut from X^k and excluded in X^{k+1}. Since the fathoming process in Steps 1 and 2 and the domain cutting process in Step 3 do not remove from X^k any feasible solution better than x_{best}, the feasible solution x_{best} must be an optimal solution to (P) when the algorithm stops at Step 3 with no integer boxes left in T^k. □

To illustrate the algorithm, let us consider a small-size numerical example:

EXAMPLE 1

$$\begin{aligned}\min\quad & f(x)=-5x_1^2-15x_1-4x_2^2-6x_2-2x_3^2-4x_3-x_4^2-9x_4\\ & \qquad -2x_5^2-18x_5,\\ \text{s.t.}\quad & g(x)=7x_1+x_2+5x_3+4x_4+2x_5\leqslant 47.5,\\ & x\in X=\{x\mid 0\leqslant x_j\leqslant 5,\ x_j\ \text{integer},\ j=1,2,3,4,5\}.\end{aligned}$$

The optimal solution of this problem is $x^*=(4,5,0,1,5)^T$ with $f(x^*)=-420$.

Applying Algorithm 1 to this example, it terminates at the 4th iteration with the optimal solution x^* after generating 10 integer subboxes. The following detailed description gives the iterative solution process.

Initial Iteration

Step 0. Set $l=(0,0,0,0,0)^T$, $u=(5,5,5,5,5)^T$, $X^1=\{\langle l,u\rangle\}$, $Y^1=X^1$, $Z^1=\emptyset$, $x_{\text{best}}=(0,0,0,0,0)^T$, $f_{\text{best}}=0$, $k=1$.

Iteration 1

Step 1. For box $\langle l,u\rangle$, we have

$$\begin{aligned}&x^R=(4.64,5,0,0,5)^T,\ \ L(x^R)=-445.71,\ \ x^F=(4,5,0,0,5)^T,\\ &x^I=(5,5,0,0,5)^T,\quad \bar{x}^F=(4,5,0,1,5)^T,\\ &x_{\text{best}}=\bar{x}^F=(4,5,0,1,5)^T,\qquad f_{\text{best}}=-420.\end{aligned}$$

Step 2. $T^1=\{\langle l,u\rangle\}$.

Step 3. Integer box $\langle l,u\rangle$ is chosen to partition. $Z^2=\emptyset$. Using (13),

$$Y^2 = \big(\langle l, u\rangle \backslash \langle (0,0,0,0,0)^T, (4,5,0,0,5)^T\rangle\big) \backslash \langle (5,5,0,0,5)^T,$$
$$(5,5,5,5,5)^T\rangle$$

is partitioned into 4 integer subboxes:

$$Y_1^2 = \langle (0,0,1,0,0)^T, (4,5,5,5,5)^T\rangle,\ \ Y_2^2 = \langle (0,0,0,1,0)^T, (4,5,0,5,5)^T\rangle,$$
$$Y_3^2 = \langle (5,0,0,0,0)^T, (5,4,5,5,5)^T\rangle,\ \ Y_4^2 = \langle (5,5,0,0,0)^T, (5,5,5,5,4)^T\rangle.$$

Thus, $X^2 = Y^2 \bigcup Z^2 = \{Y_1^2, Y_2^2, Y_3^2, Y_4^2\}$. Set $k=2$, goto Step 1.

Iteration 2

Step 1. (1) For box Y_1^2, we have $x^R = (3.93, 5, 1, 0, 5)^T$, $L(x^R) = -413.52 > f_{\text{best}}$, remove Y_1^2 from Y^2.
(2) For box Y_2^2, we have $x^R = (4, 5, 0, 1.13, 5)^T$, $L(x^R) = -421.87 < f_{\text{best}}$, $x^F = (4,5,0,1,5)^T$, $x^I = (4,5,0,2,5)^T$, $\bar{x}^F = x^F$.
(3) For box Y_3^2, we have $x^R = (5, 4, 0, 0, 4.25)^T$, $L(x^R) = -407 > f_{\text{best}}$, remove Y_3^2 from Y^2.
(4) For box Y_4^2, we have $x^R = (5, 5, 0, 0, 3.75)^T$, $L(x^R) = -427.5 < f_{\text{best}}$, $x^F = (5,5,0,0,3)^T$, $x^I = (5,5,0,0,4)^T$, $\bar{x}^F = x^F$.
Step 2. $T^2 = \{Y_2^2, Y_4^2\}$.
Step 3. Integer box Y_4^2 is chosen to partition. $Z^3 = \{Y_2^2\}$. Using (13),

$$Y^3 = \big(Y_4^2 \backslash \langle (5,5,0,0,0)^T, (5,5,0,0,3)^T\rangle\big) \backslash \langle (5,5,0,0,4)^T, (5,5,5,5,4)^T\rangle$$

is partitioned into 2 integer subboxes

$$Y_1^3 = \langle (5,5,1,0,0)^T, (5,5,5,5,4)^T\rangle,\ \ Y_2^3 = \langle (5,5,0,1,0)^T, (5,5,0,5,4)^T\rangle.$$

Thus, $X^3 = Y^3 \bigcup Z^3 = \{Y_2^2, Y_1^3, Y_2^3\}$. Set $k=3$, goto Step 1.

Iteration 3

Step 1. (1) For Y_1^3, we have $x^R = (5, 5, 1, 0, 1.25)^T$, $L(x^R) = -368.5 > f_{\text{best}}$, remove Y_1^3 from Y^3.
(2) For Y_2^3, we have $x^R = (5, 5, 0, 1, 1.75)^T$, $L(x^R) = -385.5 > f_{\text{best}}$, remove Y_2^3 from Y^3.
Step 2. $T^3 = \{Y_2^2\}$.
Step 3. Integer box Y_2^2 is chosen to partition. $Z^4 = \emptyset$. Using (13),

$$Y^4 = (Y_2^2 \backslash \langle (0,0,0,1,0)^T, (4,5,0,1,5)^T\rangle) \backslash \langle (4,5,0,2,5)^T,$$
$$(4,5,0,5,5)^T\rangle$$

is partitioned into 3 integer subboxes

$$Y_1^4 = \langle (0,0,0,2,0)^T, (3,5,0,5,5)^T\rangle,$$
$$Y_2^4 = \langle (4,5,0,2,0)^T, (4,5,0,5,4)^T\rangle,$$
$$Y_3^4 = \langle (4,0,0,2,0)^T, (4,4,0,5,5)^T\rangle.$$

Thus, $X^4 = Y^4 \bigcup Z^4 = \{Y_1^4, Y_2^4, Y_3^4\}$. Set $k=4$, goto Step 1.

Iteration 4

Step 1. (1) For Y_1^4, we have $x^R = (3,5,0,2.87,5)^T$, $L(x^R) = -396 > f_{\text{best}}$, remove Y_1^4 from Y^4.

(2) For Y_2^4, we have $x^R = (4,5,0,2,3.25)^T$, $L(x^R) = -376.5 > f_{\text{best}}$, remove Y_2^4 from Y^4.

(3) For Y_3^4, we have $x^R = (4,4,0,2,3.75)^T$, $L(x^R) = -355 > f_{\text{best}}$, remove Y_3^4 from Y^4.

Step 2. $T^4 = \emptyset$.

Step 3. Stop, $x_{\text{best}} = (4,5,0,1,5)^T$ is the optimal solution.

4. Computational Results

Algorithm 1 has been coded by Fortran 90 and has run on a Sun Blade 1000 workstation. The test problems in our computational experiment are of the following form:

$$\begin{aligned} \min \quad & f(x) = \sum_{j=1}^{n} (-c_j x_j^3 - d_j x_j^2 - e_j x_j), \\ \text{s.t.} \quad & g(x) = \sum_{j=1}^{n} b_j x_j \leqslant b, \\ & x \in X = \{x \mid l_j \leqslant x_j \leqslant u_j,\ x_j \text{ integer},\ j = 1, \ldots, n\}, \end{aligned}$$

where c_j, d_j, e_j and b_j are positive real number. The problem size n in our testing ranges from 200 to 1200. For each n, 20 test problems are randomly generated by a uniform distribution with $c_j \in [0,1]$, $d_j \in [1,10]$, $e_j \in [1,20]$, and $b_j \in [1,40]$. In all the test problems, $l_j = 1$, $u_j = 5$ and $b = \sum_{j=1}^{n} b_j l_j + 0.5(\sum_{j=1}^{n} b_j(u_j - l_j))$. Table 1 summarizes the numerical results, where min, max and avg stand for minimum, maximum and average, respectively.

From Table 1, we can see that the proposed linear approximation and partition method can find exact optimal solutions of large-scale concave knapsack problems with up to 1200 integer variables in reasonable computation time.

We have compared the performance of Algorithm 1 with two other existing methods in the literature which are applicable to nonconvex integer programming problems: dynamic programming method [1] and the hybrid method of Marstern and Morin [16]. Due to the separability of the objective and constraint functions, dynamic programming can be used to search for the optimal solution. Since an efficient implementation of the dynamic programming recursion requires that the coefficients b_j's be

Table 1. Numerical results of Algorithm 1

	Number of iterations			Number of subboxes			CPU seconds		
n	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
200	6	117	52	889	12871	5453	0.7	8.9	3.8
400	4	237	64	612	39476	13841	1.1	64.3	23.6
800	2	589	200	800	247477	87743	5.4	1183.6	420.6
1200	11	436	211	9513	320784	133755	105.5	3158.1	1220.0

Table 2. Comparison results with dynamic programming method

	Average CPU seconds	
n	Algorithm 1	Dynamic programming
200	3.7	23.2
400	24.9	140.0
600	83.6	451.7
800	273.8	975.3

positive integers, b_j's in the test problems are random integers from [1,10]. The comparison results are reported in Table 2 where the average CPU seconds are obtained by running 20 test problems for each n. The hybrid method of Marstern and Morin [16] is a combination of dynamic programming approach with dominance rules and branch-and-bound method. The basic idea of the method is to recursively generate the efficient feasible solutions of the problem and remove the inefficient feasible solutions by dominance rules. Branch-and-bound strategy is employed to remove incomplete solutions during the recursion. The comparison results are reported in Table 3 where the average CPU seconds are obtained by running 20 test problems for each n.

From Tables 2 and 3, it is clear that the proposed linearization and domain cut method significantly outperforms the dynamic programming method and the hybrid method of Marstern and Morin in terms of average CPU time.

Table 3. Comparison results with hybrid method

	Average CPU seconds	
n	Algorithm 1	Hybird Method
50	0.11	2.8
100	0.53	39.1
150	1.6	173.0
200	3.8	546.3

5. Concluding Remarks

We have proposed a new exact method for nonlinear concave integer knapsack problems. The method combines the linear approximation with a new domain cut strategy. The robustness and efficiency of the proposed algorithm have been witnessed from test problems of a size up to 1200 variables. Favorable comparison results with other existing methods in literature have also been observed. The proposed algorithm can be viewed as a special branch-and-bound method where the lower bound at each node coincides with the Lagrangian dual bound of the subproblem. In contrast to the branching rule in the conventional branch-and-bound procedure for integer programming where two new subproblems are generated at each node and one of them is selected to solve next, the proposed algorithm generates and evaluates at most n new nodes simultaneously at each level of the search tree, and the node fathoming is done for all nodes at the same level. By doing so, it is expected that better feasible solutions can be found in an early stage and at the same time more nodes can be removed from the search tree. Finally, we point out the proposed method can be extended to handle multiple constraints by using a surrogate constraint technique.

References

1. Bellman, R. and Dreyfus, S.E. (1962), *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ.
2. Benson, H.P. and Erenguc, S.S. (1990), An algorithm for concave integer minimization over a polyhedron, *Naval Research Logistics* 37, 515–525.
3. Benson, H.P., Erengue, S.S. and Horst, R. (1990), A note on adopting methods for continuous global optimization to the discrete case, *Annals of Operations Research* 25, 243–252.
4. Bretthauer, K.M., Cabot, A.V. and Venkataramanan, M.A. (1994), An algorithm and new penalties for concave integer minimization over a polyhedron, *Naval Research Logistics* 41, 435–454.
5. Bretthauer, K.M. and Shetty, B. (1995), The nonlinear resource allocation problem, *Operations Research* 43, 670–683.
6. Bretthauer, K.M. and Shetty, B (2002a), The nonlinear knapsack problem–algorithms and applications, *European Journal of Operations Research* 138, 459–472.
7. Bretthauer, K.M. and Shetty, B. (2002b), A pegging algorithm for the nonlinear resource allocation problem, *Computers and Operational Research* 29, 505–527.
8. Cabot, A.V. and Erengue, S.S. (1986), A branch and bound algorithm for solving a class of nonlinear integer programming problems, *Naval Research Logistics* 33, 559–567.
9. Cooper, M.W. (1981), Survey of methods of pure nonlinear integer programming, *Management Science 27, 353–361.*
10. Dantzig, G.B. (1957), Discrete variable extremum problems, *Operations Research* 5, 266–277.
11. Djerdjour, M., Mathur, K. and Salkin, H.M. (1988), A surrogate relaxation based on algorithm for a general class quadratic multi-dimensional knapsack problem, *Operations Research Letters* 7, 253–258.

12. Hochbaum, D. (1995), A nonlinear knapsack problem, *Operation Research Letters* 17, 103–110.
13. Horst, R. and Thoai, N.V. (1998), An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks, *OR Spektrum* 20, 47–53.
14. Ibaraki, T. and Katoh, N. (1988), *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Cambridge, MA.
15. Kodialam, M.S. and Luss, H. (1998), Algorithm for separable nonlinear resource allocation problems, *Operations Research* 46, 272–284.
16. Marsten, R.E. and Morin, T.L. (1978), A hybrid approach to discrete mathematical programming, *Mathematical Programming* 14, 21–40.
17. Martello, S. and Toth, P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, New York.
18. Mathur, K., Salkin, H.M. and Mohanty, B.B. (1986), A note on a general non-linear knapsack problems, *Operations Research Letters* 5, 79–81.
19. Mathur, K., Salkin, H.M. and Morito, S. (1983), A branch and search algorithm for a class of nonlinear knapsack problems, *Operation Research Letters* 2, 55–60.
20. Morin, T.L. and Marsten, R.E. (1976a), An algorithm for nonlinear knapsack problems, *Management Science* 22, 1147–1158.
21. Morin, T.L. and Marsten, R.E. (1976b), Branch and bound strategies for dynamic programming, *Operations Research 24*, 611–627.
22. Pardalos, P.M. and Kovoor, N. (1990), An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds, *Mathematical Programming* 46, 321–328.
23. Pardalos, P.M. and Rosen, J.B. (1988), Reduction of nonlinear integer separable programming problems', *International Journal of Computer Mathematics* 24, 55–64.
24. Rosen, J.B. and Pardalos, P.M. (1987), *Constrained Global Optimization: Algorithms and Applications*, Springer-Verlag, Berlin.
25. Sun, X.L. and Li, D. (2002), Optimality condition and branch and bound algorithm for constrained redundancy optimization in series systems, *Optimization Engineering* 3, 53–65.