J Glob Optim (2007) 39:127–154
DOI 10.1007/s10898-006-9128-7

ORIGINAL PAPER

Computing exact solution to nonlinear integer programming: Convergent Lagrangian and objective level cut method

D. Li · J. Wang · X. L. Sun

Received: 7 April 2006 / Accepted: 29 November 2006 / Published online: 16 January 2007

Abstract In this paper, we propose a convergent Lagrangian and objective level cut method for computing exact solution to two classes of nonlinear integer programming problems: separable nonlinear integer programming and polynomial zero-one programming. The method exposes an optimal solution to the convex hull of a revised perturbation function by successively reshaping or re-confining the perturbation function. The objective level cut is used to eliminate the duality gap and thus to guarantee the convergence of the Lagrangian method on a revised domain. Computational results are reported for a variety of nonlinear integer programming problems and demonstrate that the proposed method is promising in solving medium-size nonlinear integer programming problems.

Keywords Nonlinear integer programming · Lagrangian duality · Object level cut · Separable nonlinear integer programming · Polynomial zero-one programming

D. Li (✉)
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

J. Wang
Department of Management Science and Engineering, International Business College,
Qingdao University, Qingdao 266071, China
e-mail: jwang.qdu@gmail.com

X. L. Sun
Department of Management Science, School of Management, Fudan University,
Shanghai 200433, China
e-mail: xlsun@staff.shu.edu.cn

1 Introduction

We consider the following general class of nonlinear integer programming problems in this paper:

$$(P) \qquad \begin{aligned} &\min\ f(x),\\ &\text{s.t. } g_i(x) \le b_i, \quad i = 1,\ldots,m,\\ &\quad\ x \in X, \end{aligned}$$

where X is a finite integer set in $\mathbb{R}^n$, f and all g_i, $i = 1,\ldots,m$, are continuous functions on X. More specifically, we develop an exact solution scheme for the following two special cases of (P), the separable nonlinear integer programming problem:

$$(P_s) \qquad \begin{aligned} &\min\ f(x) = \sum_{j=1}^{n} f_j(x_j),\\ &\text{s.t. } g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j) \le b_i, \quad i = 1,\ldots,m,\\ &\quad\ x \in X = X_1 \times X_2 \times \cdots \times X_n \end{aligned}$$

and the constrained polynomial 0-1 programming problem:

$$(P_{0-1}) \qquad \begin{aligned} &\min\ f(x) = \sum_{k=1}^{q} c_k \prod_{j\in Q_k} x_j,\\ &\text{s.t. } g_i(x) = \sum_{k=1}^{q} a_{ik} \prod_{j\in Q_k} x_j \le b_i, \quad i = 1,2,\ldots,m,\\ &\quad\ x \in X = \{0,1\}^n, \end{aligned}$$

where, in (P_s), all f_j's are integer-valued functions, all g_{ij}'s are real valued functions and all X_j's are finite integer sets in $\mathbb{R}$ and, in (P_{0-1}), $Q_k \subseteq \{1,\ldots,n\}$ for $k = 1,\ldots,q$.

Integer programming has been one of the great challenges in front of the optimization research community for many years, due to an exponential growth in its computational complexity with respect to the problem dimension. It has been shown in the literature that many special cases of (P) are NP-hard [9,26,33]. Therefore, constructing an efficient exact algorithm for (P) is a challenging task.

Problem (P) possesses a nonconvex nature in many instances, e.g., concave integer programming [5,6,24] and polynomial integer programming [25]. The success of the continuous-relaxation-based branch-and-bound methods for solving integer programming problems relies on an ability to identify a global optimal solution to continuous relaxation subproblems. Certain solution schemes in global optimization, such as linear relaxation and convex underestimation [13,48,49] have been developed to compute lower bounds of the nonconvex continuous relaxation subproblems. It is noticed that algorithms such as branch-and-bound, outer-approximation and cutting plane developed for solving general mixed-integer nonlinear programming (MINLP) problems are applicable to problem (P) (see, e.g., [13,19,44,48,49]).

Problem (P_s) has a wide variety of applications, including resource allocation problems and nonlinear knapsack problems. In particular, manufacturing capacity planning, stratified sampling, production planning, and network reliability are special cases of (P_s) (see [7,26,46] and the references therein). Ibaraki and Katoh [26] summarized certain algorithms for singly constrained resource allocation problems where the objective function is convex and separable and the single constraint is of a special form of $\sum_{j=1}^{n} x_j = N$. Marsten and Morin [31] proposed a dynamic programming and branch-and-bound method for solving problem (P_s) where each f_j is nonincreasing on X_j. Bretthauer and Shetty [7,8] proposed a branch-and-bound algorithm for a special singly constrained case of (P_s) where all f_j's and g_{ij}'s are convex. Hochbaum [23] studied a singly constrained case of (P_s) where all f_j's and g_{ij}'s are convex and monotonically nonincreasing. The piecewise linear approximations of f_j and g_{ij}'s are used in Hochbaum [23] to convert the problem into a 0-1 linear integer programming problem.

Although dynamic programming is conceptually an ideal solution scheme for separable integer programming (P_s), the "curse of dimensionality" prevents its direct application to the multiple-constrained cases of (P_s) when m is large. Moreover, all the constraint functions, g_{ij}'s, are usually required to be integer-valued or rational-valued for an efficient implementation of dynamic programming method.

Surveys of the methods for constrained nonlinear 0-1 programming problems can be found in Hansen [20] and Hansen et al. [21]. The linearization method was proposed by Dantzig [10], Fortet [14,15] and Watters [50]. Various branch-and-bound methods or implicit enumeration methods were proposed in, for example, Hansen et al. [21] and the references therein. The cutting-plane method was originated in Granot and Hammer [18] for (P_{0-1}) with a linear objective function and was extensively studied in Balas and Mazzola [2,3] for general (P_{0-1}).

Following the backtrack concept of Geoffrion [16], Taha [47] extended the additive algorithm of Balas [1] for linear 0-1 programming to constrained polynomial 0-1 programming by designing a two-level solution scheme. Note that Taha's original results [47] can only deal with problem (P_{0-1}) with all c_j's nonnegative.

It is noted that a general twice-differentiable 0-1 program can be converted into a convex 0-1 program by adding suitable quadratic terms and using the fact $x_i^2 = x_i$ for $x_i \in \{0,1\}$. Therefore, problem (P_{0-1}) can be reduced to a convex 0-1 programming problem. It is also worth pointing out that semidefinite programming relaxation has been also developed for nonconvex quadratic 0-1 program which is a special case of (P_{0-1}) (see [22,35]).

The continuous versions of problem (P_{0-1}) has been studied by many researchers in the context of multilinear programming and polynomial programming. Lower bounding techniques and global optimization methods for multilinear programming were investigated by Ryoo and Sahinidis [37,38] and Sherali [41,42]. Note that polynomial function is a special case of factorable functions. McCormick [32] proposed a convex underestimating method for computing global solution to nonconvex factorable programming problem. Global optimization method based on reformulation-linearization technique was developed by Sherali and Wang [43] for nonconvex factorable programming problem.

The concept of duality plays an important role in discrete optimization. The Lagrangian relaxation methods are widely adopted in integer programming (see, e.g., [11,12,17,40]). As discussed in Li and White [30], the conventional Lagrangian dual method would fail to generate an optimal solution to (P) due to the existence of a duality gap. Using group theory, Bell and Shapiro [4] proposed a convergent Lagrangian duality theory for linear integer programming in which the duality gap is reduced by reshaping the feasible region. Recently, the duality gap in general nonlinear integer programming was examined and its related properties were studied in Li and Sun [28] and Li and White [30]. Nonlinear Lagrangian formulations are proposed in Li and Sun [28], Li and White [30] and Sun and Li [45] to offer a success guarantee for the dual search in generating an optimal solution of the primal integer programming problem. Although the nonlinear Lagrangian formulations possess strong duality or asymptotic strong duality, it does not lead to a decomposability which is crucial for an efficient implementation of a dual scheme.

This paper aims to develop a novel convergent Lagrangian method for (P) which is an exact solution scheme and is efficient in implementation. The proposed method exposes an optimal solution of (P) to the convex hull of the revised perturbation function by successively using objective cuts. The algorithm starts with a lower bound derived from the dual value by the conventional dual search and an upper bound by a feasible solution generated in the dual search (if any). The lower level cut and upper level cut are imposed to (P) such that the duality bound (duality gap) is forced to shrink. For (P_s), the Lagrangian relaxation with an objective cut retains the decomposability of (P_s), more specifically, results in a singly constrained separable integer programming problem which can be solved efficiently by dynamic programming. For (P_{0-1}), the Lagrangian relaxation with an objective cut results in a singly constrained polynomial zero-one programming problem which can be efficiently solved by a revised version of Taha's method proposed in this paper. The objective cut is updated successively with the distance between the upper cut and the lower cut monotonically decreasing. The algorithm terminates in a finite number of iterations, either reaching an optimal solution to (P) or reporting an infeasibility of (P).

The paper is organized as follows. We introduce some preliminary results of Lagrangian duality in Sect. 2. The idea of adopting an objective level cut in reducing the duality gap is motivated in Sect. 3 and a corresponding solution scheme is developed. In Sects. 4 and 5, the details of the algorithms and their computational implementation for (P_s) and (P_{0-1}), are described, respectively. Computational results are reported in Sect. 6 for several classes of medium-size test problems. Finally, a brief concluding remark is given in Sect. 6.

2 Lagrangian duality

2.1 Lagrangian dual formulation

By associating with each constraint in (P) a nonnegative λ_i, the Lagrangian relaxation of (P) is formulated as

$$(L_\lambda) \qquad d(\lambda) = \min_{x \in X} L(x, \lambda),$$

where $\lambda = (\lambda_1, \ldots, \lambda_m)^T \in \mathbb{R}^m_+$ and the Lagrangian function of (P) is defined as:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i (g_i(x) - b_i).$$

For separable problem (P_s), one of the prominent features of adopting the Lagrangian relaxation problem (L_λ) is that it can be decomposed into n one-dimensional problems:

$$\begin{aligned} &\min\ f_j(x_j) + \sum_{i=1}^{m} \lambda_i g_{ij}(x_j), \\ &\text{s.t. } x_j \in X_j. \end{aligned} \tag{1}$$

Notice that (1) is a problem of minimizing a univariate function over a finite integer set and its optimal solution set can be easily identified. Denote by $v(\cdot)$ the optimal value of problem $(\cdot)$. Let the feasible region and the optimal value of (P) be defined as,

$$\begin{aligned} &S = \{x \in X \mid g_i(x) \le b_i,\ i = 1, \ldots, m\}, \\ &f^* = v(P) = \min_{x \in S} f(x). \end{aligned}$$

Since $d(\lambda) \le f(x)$ for all $x \in S$ and $\lambda \ge 0$, the dual value $d(\lambda)$ always provides a lower bound for the optimal value of (P) (weak duality):

$$f^* \ge d(\lambda), \quad \forall \lambda \ge 0.$$

We assume in the sequel that $\min_{x\in X} f(x) < f^*$, otherwise $\min_{x \in X} = f^*$ must hold and (P) reduces to an unconstrained integer programming problem. The Lagrangian dual problem of (P) is to search for an optimal multiplier vector $\lambda^* \in \mathbb{R}^m_+$ which maximizes $d(\lambda)$ for all $\lambda \ge 0$:

$$(D) \qquad d(\lambda^*) = \max_{\lambda \ge 0} d(\lambda).$$

By weak duality, $f^* \ge d(\lambda^*)$ holds. The difference $f^* - d(\lambda^*)$ is called the *duality gap* between (P) and (D). Let UB be an upper bound of f^*. We denote UB $- d(\lambda^*)$ as a *duality bound* between (P) and (D). It is clear that a duality bound is always larger than or equal to the duality gap. If $f^* = d(\lambda^*)$, then the strong duality is said to be satisfied. Unfortunately, the strong duality is rarely present in integer programming.

A vector $\lambda^* \ge 0$ is said to be an *optimal generating multiplier* vector of (P) if an optimal solution x^* to (P) can be generated by solving (L_λ) with $\lambda = \lambda^*$. A pair (x^*, λ^*) is said to be an *optimal primal-dual pair* of (P) if the optimal dual solution λ^* to (D) is an optimal generating multiplier vector for an optimal solution x^* to (P). As discussed in Li and White [30], the conventional Lagrangian dual method would fail to generate an optimal solution of the primal problem (P) in two critical situations. The first situation occurs where no solution of (P) can be generated by problem (L_λ) for any $\lambda \ge 0$. The second situation occurs where no solution to problem (L_{λ^*}), with λ^* being an optimal solution to (D), is a solution to (P). The first situation mentioned above is associated with the existence of an optimal generating Lagrangian multiplier vector, while the second is related to the existence of an optimal primal-dual pair.

2.2 Perturbation function and duality gap

Let $g(x) = (g_1(x), \ldots, g_m(x))^T$ and $b = (b_1, \ldots, b_m)^T$. The perturbation function of (P) is defined as follows for $y \in \mathbb{R}^m$,

$$w(y) = \min\{f(x) \mid g(x) \le y,\ x \in X\}, \tag{2}$$

where the domain of w is

$$Y = \{y \in \mathbb{R}^m \mid \text{there exists } x \in X \text{ such that } g(x) \le y\}.$$

Note that Y is not always a convex set. The perturbation function w can be extended to the convex hull of Y by defining $w(y) = +\infty$ for $y \in \text{conv}(Y)\backslash Y$. Furthermore, w is a nonincreasing and piecewise constant ($+\infty$) function of y on conv(Y). By definition (2), $w(g(x)) \leq f(x)$ for any $x \in X$ and $w(b) = f^*$. In a process of increasing y, if there is a new point $\tilde{x} \in X$ such that $f(\tilde{x}) < w(y)$ for any $y \in \{z \in Y \mid z \leq g(\tilde{x}),\ z \neq g(\tilde{x})\}$, the perturbation function w has a downward jump at $y = g(\tilde{x})$. The point $g(\tilde{x})$ corresponding to this new point $\tilde{x}$ is called a *corner point* of the perturbation function w in the y space. Since f and g_i's are continuous functions and X is a finite integer set, there is only a finite number of corner points, say K corner points, $c_1, c_2, \ldots, c_K$. Let $f_i = w(c_i), i = 1, \ldots, K$. Define the sets of corner points in the y space and the $\{y, w(y)\}$ space, respectively, as follows,

$$C = \{c_i = (c_{i1}, c_{i2}, \ldots, c_{im}) \mid i = 1, \ldots, K\},$$
$$\Phi_c = \{(c_i, f_i) \mid i = 1, \ldots, K\}.$$

It is clear that $(y, w(y)) \in \Phi_c$ iff for any $z \in Y$ satisfying $z \leq y$ and $z \neq y$, it holds $w(z) > w(y)$.

From the definition of the corner point, the domain Y can be decomposed into K subsets with each c_i as the lower end of the corresponding subset Y_i. More specifically, we have $Y = \cup_{i=1}^K Y_i$ with $c_{ij} = \min\{y_j \mid y \in Y_i\}, j = 1, \ldots, m$, and w takes a constant f_i over Y_i:

$$w(y) = f_k, \quad \forall y \in Y_k,\ k = 1, \ldots, K.$$

Define the *convex envelope function* of w to be the greatest convex function majorized by w:

$$\psi(y) = \max\{h(y) \mid h(y) \text{ is convex on } Y,\ h(y) \leq w(y),\ \forall y \in Y\}.$$

It can be easily seen that ψ is piecewise linear and nonincreasing on Y and $w(y) \geq \psi(y)$ for all $y \in Y$. Since ψ is convex and nonincreasing, we have

$$\psi(y) = \max\{-\lambda^T y + r \mid \lambda \in \mathbb{R}^m_+,\ r \in \mathbb{R}, \text{ and } -\lambda^T z + r \leq w(z),\ \forall z \in Y\}$$

or equivalently,

$$\begin{aligned} \psi(y) = \max\ & -\lambda^T y + r, \\ \text{s.t. }\ & -\lambda^T c_k + r \leq f_k, \quad k = 1, \ldots, K, \\ & \lambda \in \mathbb{R}^m_+, \quad r \in \mathbb{R}. \end{aligned} \tag{3}$$

For any fixed $y \in Y$, introduce a dual variable $\mu_k \geq 0$ for each constraint $-\lambda^T c_k + r \leq f_k,\ k = 1, \ldots, K$. Dualizing the linear program (3) then yields

$$\begin{aligned} \psi(y) = \min\ & \sum_{k=1}^K \mu_k f_k, \\ \text{s.t. }\ & \sum_{k=1}^K \mu_k c_k \leq y, \\ & \sum_{i=1}^K \mu_k = 1,\ \mu_k \geq 0, \quad k = 1, \ldots, K. \end{aligned} \tag{4}$$

The following result establishes the relation between the duality and the perturbation function.

Theorem 1 ([27,29]) *Let (λ^*, r^*) and μ^* be optimal solutions to* (3) *and* (4) *with $y = b$, respectively. Then*

(1) *λ^* is an optimal solution to the dual problem (D) and*

$$\psi(b) = \max_{\lambda \geq 0} d(\lambda) = d(\lambda^*).$$

(2) *For each k with $\mu_k^* > 0$, any $\bar{x} \in X$ satisfying $(g(\bar{x}), f(\bar{x})) = (c_k, f_k)$ is an optimal solution to the Lagrangian problem (L_{λ^*}).*

3 Objective level cut method

Stimulated by the relationship between the duality gap and the geometry of the perturbation function, we develop in this section the solution scheme of the convergent Lagrangian and objective level cut algorithm for (P).

3.1 Motivation

To motivate the solution algorithm, let us consider the following example:

Example 1

$$\begin{aligned} \min\ & f(x) = -2x_1^2 - x_2 + 3x_3^2, \\ \text{s.t.}\ & 5x_1 + 3x_2^2 - \sqrt{3}x_3 \leq 7, \\ & x \in X = \{x \in \mathbb{Z}^2 \mid 0 \leq x_i \leq 2,\ i = 1, 2, 3\}. \end{aligned}$$

The optimal solution of this example is $x^* = (1, 0, 0)^T$ with $f^* = f(x^*) = -2$. The perturbation function of this problem, $w(\cdot)$, and its convex envelope function, $\psi(\cdot)$, are illustrated in Fig. 1. From Fig. 1 we can see that point C that corresponds to the optimal solution x^* "hides" above the convex envelope of the perturbation function and therefore there is no optimal generating multiplier for x^*. In other words, it is impossible for x^* to be found by the conventional Lagrangian dual method.

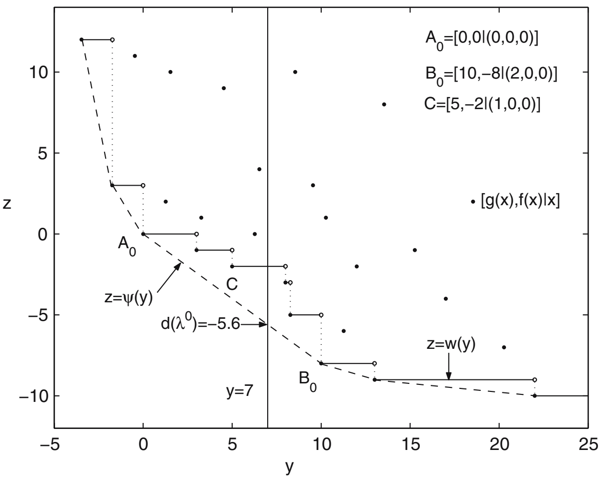

Fig. 1 The perturbation function of Example 1

The optimal solution to (D) is $\lambda^0 = 0.8$ with $d(\lambda^0) = -5.6$. Thus, the duality gap is $f(x^*) - d(\lambda^0) = -2 + 5.6 = 3.6$. A key observation is that point C can be exposed to the convex envelope or the convex hull of the perturbation function by adding an objective cut. As a matter of fact, since A_0 corresponds to a feasible solution $x^0 = (0, 0, 0)^T$, the function value $f(x^0) = 0$ is an upper bound of f^*. Moreover, by the weak duality, the dual value $d(\lambda^0) = -5.6$ is a lower bound of f^*. The current duality bound is $0 - (-5.6) = 5.6$. Therefore, adding an objective cut of $-5.6 \le f(x) \le 0$ to the original problem does not exclude the optimal solution while the perturbation function will be reshaped, or more precisely, be re-confined. Since the objective function is integer-valued, we can set a stronger objective cut of $-5 \le f(x) \le -1$ after storing the current best feasible solution x^0 as the incumbent. The modified problem then has the following form:

$$\begin{aligned} \min\ & f(x) = -2x_1^2 - x_2 + 3x_3^2, \\ \text{s.t.}\ & 5x_1 + 3x_2^2 - \sqrt{3}x_3 \le 7, \\ & x \in X_1 = X \cap \{x \mid -5 \le f(x) \le -1\}. \end{aligned} \tag{5}$$

The perturbation function of problem (5) is shown in Fig. 2. The optimal dual multiplier to (5) is $\lambda^1 = 0.7593$ with dual value $d(\lambda^1) = -4.0372$. Since $x^1 = (0, 1, 0)^T$ corresponding to A_1 is feasible, the duality bound is now reduced to $f(x^1) - (-4.0372) = -1 + 4.037 = 3.0372$. Again we can add an objective cut $-4 \le f(x) \le f(x^1) - 1 = -2$ to (5) and obtain the following problem:

$$\begin{aligned} \min\ & f(x) = -2x_1^2 - x_2 + 3x_3^2, \\ \text{s.t.}\ & 5x_1 + 3x_2^2 - \sqrt{3}x_3 \le 7, \\ & x \in X_2 = X \cap \{x \mid -4 \le f(x) \le -2\}. \end{aligned} \tag{6}$$

The perturbation function of problem (6) is shown in Fig. 3. The optimal dual multiplier is $\lambda^2 = 0.3333$ with dual value $d(\lambda^2) = -2.6667$. Now point C corresponding to x^* is exposed to the convex hull of the perturbation function and the duality bound is reduced to $f(x^*) - (-2.6667) = -2 + 2.6667 = 0.6667 < 1$. Since the objective function is integer-valued, we claim that $x^* = (1, 0, 0)^T$ is the optimal solution to the original problem.

This example clearly illustrates a procedure of gradually reducing the duality bound and thus eventually eliminating the duality gap by using objective cuts. The convergent Lagrangian and objective level cut method exposes an optimal solution of (P) to the convex hull of the revised perturbation function by successively using objective cuts. The algorithm starts with a lower bound derived from the dual value by the conventional dual search and an upper bound by a feasible solution generated in the dual search (if any). The lower level cut and upper level cut are imposed to (P) such that the duality bound (duality gap) is forced to shrink. The objective cut is updated successively with the distance between the upper cut and the lower cut monotonically decreasing. The algorithm terminates in a finite number of iterations, either reaching an optimal solution to (P) or reporting an infeasibility of (P).

We can make the following clear statement: with a price of complicating the conventional Lagrangian relaxation of (P) by attaching a constraint to confine the optimal value within a certain range, we are able to remove the notorious phenomenon of duality gap associated with the conventional Lagrangian dual method, thus guaranteeing the identification of an exact solution to (P).

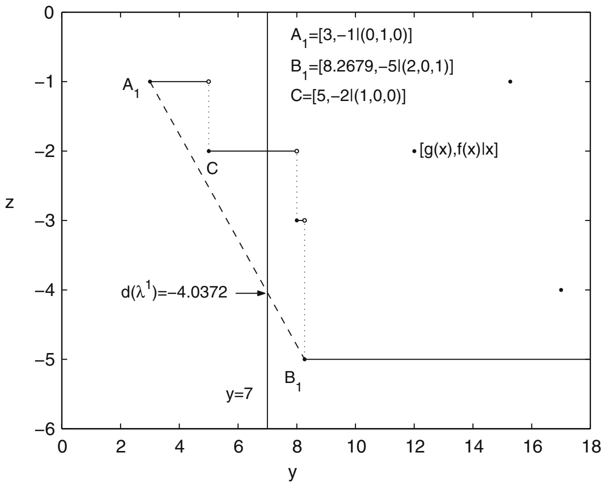

Fig. 2 The perturbation function of problem (5)

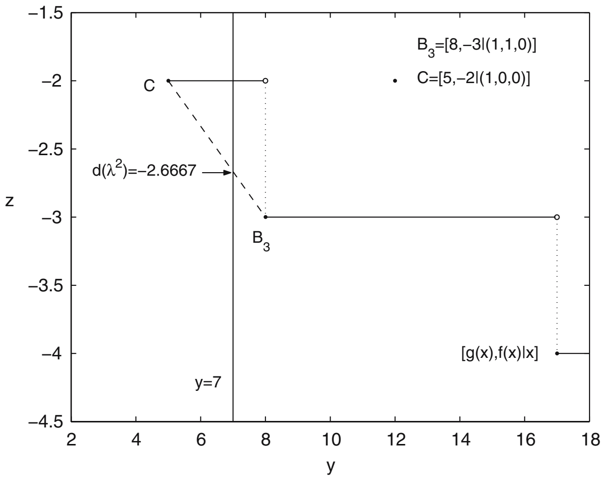

Fig. 3 The perturbation function of problem (6)

3.2 Objective level cut scheme

Consider the following modified version of (P) by imposing a lower cut l and an upper cut u:

$$
\begin{aligned}
(P(l,u)) \qquad & \min\ f(x), \\
& \text{s.t. } g_i(x) \le b_i, \quad i = 1,\dots,m, \\
& \quad x \in X(l,u) = \{x \in X \mid l \le f(x) \le u\}.
\end{aligned}
$$

It is obvious that $(P(l,u))$ is equivalent to (P) if $l \le f^* \le u$. The Lagrangian relaxation of $(P(l,u))$ is:

$$(L_\lambda(l,u)) \qquad d(\lambda,l,u) = \min_{x\in X(l,u)} L(x,\lambda),$$

where $\lambda \in \mathbb{R}^m_+$ and $L(x, \lambda) = f(x) + \sum_{i=1}^m \lambda_i(g_i(x) - b_i)$. The Lagrangian dual problem of $(P(l, u))$ is then given as

$$(D(l,u)) \quad \max_{\lambda \in \mathbb{R}^m_+} d(\lambda, l, u).$$

Since $d(\lambda)$ or $d(\lambda, l, u)$ is a nonsmooth concave function of λ, the subgradient method or the outer Lagrangian linearization method (see, e.g., [11,34,40]) can be used to solve (D) and $(D(l, u))$. In practice, the subgradient method terminates at an approximate solution when certain stopping criteria are met. Next, we discuss the properties of $(P(l, u))$ and its dual $(D(l, u))$.

Lemma 1

(1) *Let $\lambda^*(l, u)$ denote the optimal solution to $(D(l, u))$. The optimal dual value $d(\lambda^*(l, u), l, u)$ is a nondecreasing function of l.*
(2) *If $l \le f^* \le u$, then $d(\lambda^*) \le d(\lambda^*(l, u), l, u) \le f^*$. Moreover, let $\sigma = \max\{f(x) \mid f(x) < f^*, x \in X \setminus S\}$. If $\sigma < l \le f^*$, then $\lambda^*(l, u) = 0$ and $d(\lambda^*(l, u), l, u) = f^*$.*
(3) *For $l < f^*$, we have $d(\lambda^*(l, u), l, u) \ge l$.*

Proof

(1) If $l_1 \le l_2$, then $d(\lambda, l_1, u) \le d(\lambda, l_2, u)$ for all $\lambda \in \mathbb{R}^m_+$. Thus,

$$d(\lambda^*(l_1, u), l_1, u) = \max_{\lambda \in \mathbb{R}^m_+} d(\lambda, l_1, u) \le \max_{\lambda \in \mathbb{R}^m_+} d(\lambda, l_2, u) = d(\lambda^*(l_2, u), l_2, u).$$

(2) Since $X(l, u) \subseteq X$, we have

$$d(\lambda) = \min_{x \in X} L(x, \lambda) \le \min_{x \in X(l,u)} L(x, \lambda) = d(\lambda, l, u), \quad \forall \lambda \in \mathbb{R}^m_+.$$

Thus, $d(\lambda^*) \le d(\lambda^*(l, u), l, u)$. If $l \le f^* \le u$, then $S^* \subseteq X(l, u)$, where S^* is the set of optimal solutions to (P). For any $\lambda \in \mathbb{R}^m_+$, we have

$$\begin{aligned} d(\lambda, l, u) &= \min_{x \in X(l,u)} L(x, \lambda) \\ &\le \min_{x \in S^*} L(x, \lambda) \\ &\le \min_{x \in S^*} f(x) \\ &= f^*. \end{aligned}$$

Therefore $d(\lambda^*(l, u), l, u) \le f^*$. Suppose that $\sigma < l \le f^* \le u$, then there is no infeasible point x in $X(l, u)$ with $f(x) < f^*$. Thus

$$d(0, l, u) = \min_{x \in X(l,u)} f(x) = \min_{x \in S^*} f(x) = f^* \ge \min_{x \in S^*} L(x, \lambda) \ge d(\lambda, l, u)$$

for all $\lambda \in \mathbb{R}^m_+$. Thus, $\lambda = 0$ solves $(D(l, u))$ and $d(0, l, u) = f^*$.
(3) Consider the perturbation function of $(P(l, u))$. The set of corner points of it is a subset of Φ_c satisfying $l \le f_k \le u$. Thus, applying (4) and Theorem 1, we infer that there exist an index set $I(l, u) \subset \{1, 2, \ldots, K\}$ and $\mu^*_k(l, u) > 0$, $k \in I(l, u)$, such that

$$
\begin{aligned}
& d(\lambda^*(l,u),l,u) = \sum_{k\in I(l,u)} \mu_k^*(l,u) f_k, && (7)\\
& \sum_{k\in I(l,u)} \mu_k^*(l,u) c_k \le b,\\
& \sum_{k\in I(l,u)} \mu_k^*(l,u) = 1,\\
& l \le f_k \le u,\ k \in I(l,u).
\end{aligned}
$$

Since for each $k \in I(l,u)$, $f_k \ge l$, the above conditions imply that $d(\lambda^*(l,u),l,u) \ge l$. □

Lemma 1 reveals that the quality (the tightness) of the dual search can be improved by raising the value of the lower objective level cut.

Lemma 2 *If* $d(\lambda^*(l,u),l,u) < v(P) = f^*$, *then*

$$\min\{f(x) \mid x \in T(\lambda^*(l,u),l,u) \setminus S\} \le d(\lambda^*(l,u),l,u),$$

where $T(\lambda^*(l,u),l,u)$ *is the solution set to problem* $(L_\lambda(l,u))$ *with* $\lambda = \lambda^*(l,u)$.

Proof From (7), we have

$$\sum_{k\in I(l,u)} \mu_k^*(l,u)(f_k - d(\lambda^*(l,u),l,u)) = 0.$$

If there is a k such that f_k is not equal to $d(\lambda^*(l,u),l,u)$, then there must be a k_1 such that f_{k_1} is strictly greater than $d(\lambda^*(l,u),l,u)$ and there must be a k_2 such that f_{k_2} is strictly smaller than $d(\lambda^*(l,u),l,u)$. From the weak duality, the solution corresponding to f_{k_2} must be infeasible in (P). If all f_k's are equal to $d(\lambda^*(l,u),l,u)$, then all solutions in $T(\lambda^*(l,u),l,u)$ must be infeasible from the assumption of $d(\lambda^*(l,u),l,u) < f^*$. □

Lemma 2 implies that at least one infeasible solution will be removed when placing a cut higher than $d(\lambda^*(l,u),l,u)$.

One crucial issue in efficiently implementing this solution idea is how to solve $(L_\lambda(l,u))$, the relaxation problem of the revised problem $(P(l,u))$ (such as the Lagrangian relaxations in (5) and (6)). In the following two sections, we will discuss the solution schemes for (P_s) and (P_{0-1}), respectively.

4 Algorithm for separable nonlinear integer programming

In this section, we describe the details of the algorithm for separable nonlinear integer programming problem (P_s). Notice that, for problem (P_s), $L(x,\lambda) = \sum_{j=1}^n \theta_j(x_j,\lambda) - \alpha(\lambda)$ where $\theta_j(x_j,\lambda) = f_j(x_j) + \sum_{i=1}^m \lambda_i g_{ij}(x_j)$ and $\alpha(\lambda) = \sum_{i=1}^m \lambda_i b_i$. Thus, problem $(L_\lambda(l,u))$ can be explicitly written for (P_s) as:

$$
\begin{aligned}
d(\lambda,l,u) = \min\ & \sum_{j=1}^n \theta_j(x_j,\lambda) - \alpha(\lambda),\\
\text{s.t.}\ & l \le \sum_{j=1}^n f_j(x_j) \le u,\\
& x \in X.
\end{aligned}
$$

It is clear that for (P_s) the Lagrangian relaxation problem $(L_\lambda(l,u))$ is a separable integer programming problem with a lower bound and upper bound constraint for $f(x)$. As each $f_j(x_j)$ is assumed to be integer-valued for all $x_j \in X_j$ in (P_s), $(L_\lambda(l,u))$ can be then efficiently solved by dynamic programming. Let

$$s_k = \sum_{j=1}^{k-1} f_j(x_j), \quad k = 2, \ldots, n+1$$

with an initial condition $s_1 = 0$. Then $(L_\lambda(l,u))$ can be solved by the following dynamic programming formulation:

$$\begin{aligned}
\text{(DP)} \qquad & \min\ s_{n+1} + \sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i g_{ij}(x_j), \\
& \text{s.t. } s_{j+1} = s_j + f_j(x_j), \quad j = 1, 2, \ldots, n, \\
& \quad s_1 = 0, \\
& \quad l \le s_{n+1} \le u, \\
& \quad x_j \in X_j, \quad j = 1, 2, \ldots, n.
\end{aligned}$$

The state in the above dynamic programming formulation takes finite values at each stage. All the solutions to $(L_\lambda(l,u))$ can be generated using the conventional dynamic programming technique.

We now describe the algorithm for (P_s) as follows.

Algorithm 1 (*Convergent Lagrangian and objective level cut method for* (P_s))

Step 0 (*Initialization*)

(1) Solve the dual problem (D) by using the subgradient method or by the outer Lagrangian linearization method. Let λ^0 be the best dual vector found. Set $d^0 = d(\lambda^0)$.

(2) Let x^* denote the current best feasible solution (if there is one) and set $v^0 = f(x^*)$. The initial feasible solution can either be found during the dual search or by certain heuristic method. If $v^0 - d^0 < 1$, stop and x^* is an optimal solution to (P_s); Otherwise, set $l_0 = \lceil d^0 \rceil$, $u_0 = v^0 - 1$ and $k = 0$, where $\lceil x \rceil$ is the minimum integer number larger than or equal to x.

(3) When no feasible solution is found, set v^0 to be equal to an upper bound of $f(x)$ over X. If $v^0 - d^0 < 0$, stop and there is no feasible solution to (P_s); Otherwise, set $l_0 = \lceil d^0 \rceil$, $u_0 = v^0$ and $k = 0$.

Step 1 (*finding feasible solution*). If $l_k = u_k$, goto Step 3. Otherwise, solve the following problem using dynamic programming,

$$\begin{aligned}
(P_f) \qquad & \min\ g_{\lambda^k}(x) = \sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i^k g_{ij}(x_j), \\
& \text{s.t. } l_k \le f(x) \le u_k, \\
& \quad x \in X.
\end{aligned}$$

Let C^k be the set of optimal solutions to the above problem.

(1) If there is a feasible solution in C^k, then set the incumbent $x^* = \arg\min\{f(x) \mid x \in C^k \cap S\}$ and $v^k = f(x^*)$, where S is the feasible region of (P_s). If $v^k - l_k < 1$, stop and the current incumbent x^* is an optimal solution to (P_s). Otherwise, set $u_{k+1} = v^k - 1$, $l_{k+1} = l_k$, $\lambda^{k+1} = \lambda^k$ and $k := k + 1$, and goto Step 1.

(2) If for any $x \in C^k, g_{\lambda^k}(x) > \sum_{i=1}^m \lambda_i^k b_i$ holds, stop. The current incumbent x^* is an optimal solution to (P_s) or there is no feasible solution to (P_s) if no incumbent has been found.

Step 2 (*dual search with objective cut*). Solve $(D(l_k, u_k))$ by the subgradient method or the outer Lagrangian linearization method, while the Lagrangian relaxation problem $(L_\lambda(l_k, u_k))$ is solved by using dynamic programming. The subgradient method terminates when the algorithm is not able to increase the dual value after a given number of iterations. Let λ^k be the dual vector that generates the highest dual value in the dual search process. Set $d^k = d(\lambda^k, l_k, u_k)$.

(1) If there is a feasible solution x^* found during the dual search process, replace the incumbent by x^*, set $v^k = f(x^*)$, $u_{k+1} = v^k - 1$, $l_{k+1} = \max\{l_k, \lceil d^k \rceil\}$, $k := k + 1$, and goto Step 1.

(2) If no feasible solution is found and $d^k > l_k$, set $l_{k+1} = \lceil d^k \rceil$, $u_{k+1} = u_k$, $k := k + 1$, and goto Step 1.

Step 3 (*finding feasible solution when* $\lambda = 0$). Solve the following dynamic programming problem

$$
(\mathrm{DP}_0) \quad \begin{aligned} \min\ & s_{n+1}, \\ \text{s.t.}\ & s_{j+1} = s_j + f_j(x_j), \quad j = 1, 2, \ldots, n, \\ & s_1 = 0, \\ & l_k \le s_{n+1} \le u_k, \\ & x_j \in X_j, \quad j = 1, 2, \ldots, n. \end{aligned}
$$

(1) If there is a feasible optimal solution x^* to (DP_0), stop. The incumbent x^* is the optimal solution to (P_s).

(2) Set $u_{k+1} = u_k$, $l_{k+1} = v(\mathrm{DP}_0) + 1$. If $v^k - l_{k+1} < 1$, stop. The incumbent x^* is an optimal solution to (P_s) or there is no feasible solution to (P_s) if no incumbent has been found. Otherwise, set $k := k + 1$ and goto Step 1.

Step 1 in the above algorithm is adopted to speed up the convergence of the algorithm. When the objective level cut is updated, solving (P_f) could sometimes identify a feasible solution of (P_s) with an objective level less than u_k. There exist multiply constrained cases where more than one points $(g(x), f(x))$ with $g(x) \not\le b$ surrounding the axis $y = b$ and span a horizontal plane (corresponding to $\lambda = 0$) with the same f value (being the lowest objective value over the defined domain). In such a situation, the dual search method will fail to raise the dual value higher than the lowest objective value [29]. Step 3 of the above algorithm deals with such a kind of situations.

Theorem 2 *Algorithm 1 either finds an optimal solution of (P_s) or reports an infeasibility of (P_s) in at most $u_0 - l_0 + 1$ iterations.*

Proof First, from the algorithm and Lemma 1, it always holds $l_k \le f^*$. It is clear that (P_s) is infeasible if the algorithm stops at Step 0 (3), Step 1 (2) or Step 3 (2) when the incumbent is empty. The optimality of the incumbent x^* is obvious when the algorithm stops at Step 0 (2) or Step 1 (1). If the algorithm stops at Step 1 (2), then there is no feasible solution x satisfying $l_k \le f(x) \le u_k$. Thus, from the algorithm, if the incumbent is x^*, then $f(x^*) = u_k + 1$ and x^* is an optimal solution to (P_s). If the algorithm stops at Step 3 (1), then $\lambda = 0$ is the dual optimal solution to $(P(l_k, u_k))$ and $f(x^*) \ge l_k$ and thus x^* must be an optimal solution to (P_s). If the algorithm stops at Step 3 (2), then there is no feasible solution x satisfying $l_k \le f(x) \le u_k$ and the stopping condition $v^k - l_{k+1} < 1$ implies that there is no better feasible solution than the incumbent x^*.

Suppose that the algorithm does not stop at iteration k, then by the algorithm, either $u_{k+1} \le u_k - 1$ or $l_{k+1} \ge l_k + 1$. Notice that for any k, $l_k \le f^* \le u_k + 1$ holds. Therefore, in at most $u_0 - l_0$ iterations, $u_k = l_k$ will be satisfied. If the algorithm does not stop before $u_0 - l_0 + 1$ iterations, then the algorithm will stop in $(u_0 - l_0 + 1)$th iteration either at Step 3 (1) or at Step 3 (2), reaching an optimal solution or reporting an infeasibility of (P_s). □

We now discuss several implementation issues of the dynamic programming in Algorithm 1. Three techniques are developed to facilitate an efficient use of dynamic programming: partition of the objective level cut, reduction of the state space and feasibility check of (DP_0).

The initial duality bound $u_0 - l_0$ at Step 0 of Algorithm 1 has a great effect on the efficiency of dynamic programming when solving $(P(l_k, u_k))$. As a matter of fact, if the initial duality bound is very large, then the dynamic programming can be very time-consuming and inefficient due to a large range of the state space. In order to reduce the range without losing any optimal solution, a partition scheme of the objective level cut is proposed to divide the range $[l_0, u_0]$ at Step 0 into q smaller nonoverlapping blocks such that

$$[l_0, u_0] = \cup_{t=1}^{q} [l_0^t, u_0^t],$$

where $l_0^1 = l_0$, $u_0^q = u_0$, and $l_0^{t+1} = u_0^t + 1$. The original problem can be then divided into q subproblems with $t = 1, 2, \ldots, q$:

$$\begin{aligned} (P^t) \qquad & \min\ f(x), \\ & \text{s.t. } g_i(x) \le b_i, \quad i = 1, \ldots, m, \\ & \quad l_0^t \le f(x) \le u_0^t,\ x \in X. \end{aligned}$$

These q problems will be solved successively from $t = 1$ to $t = q$. If an optimal solution x^* is found in problem (P^t) for $1 \le t \le q$, then x^* is also an optimal solution to (P_s) and there is no need to solve $(P^{t+1}), \ldots, (P^q)$. If all problems (P^t) are infeasible, then we claim that the original problem is infeasible.

Next, we discuss the strategy for reducing state space. Let $\bar{s}_j$, $\underline{s}_j$ denote the upper bound and lower bound of the range of state variable s_j, respectively. Let

$$\bar{f}_j = \max_{l_j \le x_j \le u_j} f_j(x_j),$$
$$\underline{f}_j = \min_{l_j \le x_j \le u_j} f_j(x_j).$$

With the initial condition $\bar{s}_1^F = \underline{s}_1^F = 0$, the range s_j^F of the state variable s_j at stage j can be determined by a forward recursive formulation,

$$\bar{s}_{j+1}^F = \bar{s}_j^F + \bar{f}_j \quad \text{for } j = 1, \ldots, n,$$
$$\underline{s}_{j+1}^F = \underline{s}_j^F + \underline{f}_j \quad \text{for } j = 1, \ldots, n.$$

With the initial condition $\bar{s}_{n+1}^B = u_k$, $\underline{s}_{n+1}^B = l_k$, the range s_j^B of the state variable s_j at stage j can be determined by a backward recursive formulation,

$$\bar{s}_j^B = \bar{s}_{j+1}^B - \underline{f}_j \quad \text{for } j = n, \ldots, 1,$$
$$\underline{s}_j^B = \underline{s}_{j+1}^B - \bar{f}_j \quad \text{for } j = n, \ldots, 1.$$

Therefore, the exact expression of the state range can be given as follows:

$$[\underline{s}_j, \bar{s}_j] = \begin{cases} [0, 0] & \text{for } j = 1, \\ [\underline{s}_j^B, \bar{s}_j^B] \cap [\underline{s}_j^F, \bar{s}_j^F] & \text{for } j = 2, \ldots, n, \\ [l^k, u^k] & \text{for } j = n+1. \end{cases} \tag{8}$$

If any $[\underline{s}_j, \bar{s}_j]$ is empty, then $P(l_k, u_k)$ has no feasible solution. In general, the state space of dynamic programming can be significantly reduced by (8).

Now we discuss the implementation of solving (DP$_0$) at Step 3 of Algorithm 1, a situation where λ is set to be zero in the dual search. Since there may exist a large number of optimal solutions to (DP$_0$), an efficient ordering of the optimal solutions by certain rules is crucial to the feasibility check process. For given $\mu \geq 0$ and $\mu \neq 0$, consider the following surrogate constraint:

$$g^\mu(x) = \sum_{i=1}^m \mu_i g_i(x) \leq \sum_{i=1}^m \mu_i b_i = b^\mu.$$

Let $S_\mu = \{x \in X \mid g^\mu(x) \leq b^\mu\}$. It is clear that $S \subseteq S_\mu$. Suppose that the set of optimal solutions to (DP$_0$) is T_0. Rank the points in T_0 from the smallest to the largest in terms of the value of $g^\mu(x)$:

$$T_0 = \{x^1, x^2, \ldots, x^N\}.$$

Let t be such that $g^\mu(x^t) \leq b^\mu$ and $g^\mu(x^{t+1}) > b^\mu$. The point x^t is called a "turning point." When solving (DP$_0$) by dynamic programming, we generate and calculate $g^\mu(x^k)$ for $k = 1, 2, \ldots$, till a feasible solution to (P) is found or a turning point is met. In the latter case there is no feasible solution in T_0. In the worst case, checking feasibility of T_0 requires generating $t + 1$ optimal solutions in T_0.

Finally, we point out that although the objective function is assumed to be integer-valued in the algorithm, a rational objective function can be also handled by multiplying a suitable number.

To illustrate Algorithm 1, we consider the following small-size example.

Table 1 Iteration process of Example 2

Iteration	λ^k	d^k	x^*	$f(x^*)$	l_k	u_k
0	$(0.853, 0, 0.915)^T$	−548.526			−548	113
1	$(0.853, 0, 0.915)^T$	−548.526	$(-1, -4, 5, 4, 5)^T$	−367	−548	−368
2	$(0.853, 0, 0.915)^T$	−548.526	$(-2, -4, 5, 4, 5)^T$	−373	−548	−374
3	$(0.853, 0, 0.915)^T$	−548.526	$(-3, -4, 5, 4, 5)^T$	−385	−548	−386
4	$(0.853, 0, 0.915)^T$	−548.526	$(-1, -5, 5, 5, 5)^T$	−400	−548	−401
5	$(0.853, 0, 0.915)^T$	−548.526	$(-2, -5, 5, 5, 5)^T$	−406	−548	−407
6	$(0.853, 0, 0.915)^T$	−548.526	$(-3, -5, 5, 5, 5)^T$	−418	−548	−419
7	$(0.246, 0, 0.385)^T$	−540.492	$(-3, -5, 5, 5, 5)^T$	−418	−540	−419
8	$(0, 0, 0)^T$	−540.000	$(-3, -5, 5, 5, 5)^T$	−418	−539	−419
9	$(0.140, 0, 0.151)^T$	−530.359	$(-3, -5, 5, 5, 5)^T$	−418	−530	−419
10	$(0, 0, 0)^T$	−530.000	$(-3, -5, 5, 5, 5)^T$	−418	−529	−419
11	$(0.047, 0, 0.047)^T$	−528.899	$(-3, -5, 5, 5, 5)^T$	−418	−528	−419
12	$(0, 0, 0)^T$	−528.000	$(-3, -5, 5, 5, 5)^T$	−418	−527	−419
13	$(0, 0, 0)^T$	−527.000	$(-4, -5, 5, 2, 5)^T$	−526	−527	−527
14	$(0, 0, 0)^T$	−527.000	$(-4, -5, 5, 2, 5)^T$	−526		

Example 2

$$\begin{aligned}
\min\ & -3x_1 - 3x_1^2 + 8x_2 - 7x_2^2 - 5x_3 - 3x_3^2 + 2x_4 + 4x_4^2 - 4x_5 - 7x_5^2,\\
\text{s.t. } & 7x_1 + 7x_1^2 + 4x_2 + 4x_2^2 - 8x_3 - 7x_3^2 - 7x_4 + 2x_4^2 - 5x_5 + 2x_5^2 \le -6,\\
& 8x_1 - 5x_1^2 + 4x_2 - 7x_2^2 - 4x_3 + 8x_3^2 + 7x_4 - 6x_4^2 - 2x_5 - 7x_5^2 \le -2,\\
& -x_1 - 3x_1^2 - 2x_2 + x_2^2 - 2x_3 + 8x_3^2 - 5x_4 - 3x_4^2 + 5x_5 - 7x_5^2 \le 9,\\
& x \in X = \{x \in \mathbb{Z}^5 \mid -5 \le x_i \le 5,\ i = 1, 2, 3, 4, 5\}.
\end{aligned}$$

It can be verified that the optimal solution of Example 2 is $x^* = (-4, -5, 5, 2, 5)^T$ with $f(x^*) = -526$.

The initial dual value is $d^0 = -548.526$ and an upper bound of $f(x)$ is $v^0 = 113$. Therefore, the initial interval of the objective level cut is $[-548, 113]$. A partition scheme is used to divide the initial interval of objective cut into smaller ones with an interval length of 200. The algorithm finds the optimal solution x^* at iteration 13. The dual search at iteration 14 finds a zero optimal dual solution and there is no feasible solution in the set of optimal solutions to the corresponding Lagrangian relaxation problem. The algorithm thus terminates and reports x^* as an optimal solution. Table 1 summaries the iteration process of the algorithm.

5 Algorithm for polynomial 0-1 programming

In this section, we first present a two-level solution scheme for the Lagrangian relaxation of the polynomial 0-1 programming problem (P_{0-1}). The main algorithm for (P_{0-1}) is then proposed with an illustrative numerical example.

5.1 Solution scheme for Lagrangian relaxation problem

For polynomial 0-1 programming problem (P_{0-1}), the Lagrangian relaxation of $(P(l,u))$ becomes the following singly constrained polynomial 0-1 programming problem if we set the upper cut u at infinity,

$$(L_\lambda(l)) \qquad d(\lambda,l) = \min_{x\in\{0,1\}^n} \sum_{k=1}^{q} c_k \prod_{j\in Q_k} x_j + \sum_{i=1}^{m} \lambda_i \left[\sum_{k=1}^{q} a_{ik} \prod_{j\in Q_k} x_j - b_i \right],$$
$$\text{s.t. } l \le f(x) = \sum_{k=1}^{q} c_k \prod_{j\in Q_k} x_j,$$
$$x \in \{0,1\}^n.$$

5.1.1 *Two-level reformulation*

Similar to Taha [47], problem $(L_\lambda(l))$ can be converted into the following equivalent two-level formulation that consists of a linear 0-1 master program

$$\begin{aligned} \min\ & \tilde f(y) = \sum_{k=1}^{q} \tilde a_k y_k, \\ \text{s.t. } & \tilde g(y) = \sum_{k=1}^{q} \tilde c_k y_k \le \tilde b, \\ & y_k \in \{0,1\}, \quad k = 1,2,\ldots,q \end{aligned} \tag{9}$$

and a set of nonlinear secondary constraints

$$y_k = \begin{cases} \prod_{j\in Q_k} x_j & k \in J^+ = \{k \mid c_k + \sum_{i=1}^m \lambda_i a_{ik} \ge 0\}, \\ 1 - \prod_{j\in Q_k} x_j & k \in J^- = \{k \mid c_k + \sum_{i=1}^m \lambda_i a_{ik} < 0\}, \end{cases} \tag{10}$$

where $\tilde a_k = c_k + \sum_{i=1}^m \lambda_i a_{ik}$ and $\tilde c_k = -c_k$ for $k \in J^+$, $\tilde a_k = -c_k - \sum_{i=1}^m \lambda_i a_{ik}$ and $\tilde c_k = c_k$ for $k \in J^-$, $\tilde b = -l + \sum_{k\in J^-} c_k$.

5.1.2 *Partial solutions*

Let $N = \{1,\ldots,n\}$, $M = \{1,\ldots,m\}$, and $Q = \{1,\ldots,q\}$. Following the backtrack scheme in Geoffrion [16], let $I_t \subseteq Q$ denote the index set of y_k's determined at iteration t. Define a signed index set

$$J_t = \{\xi \mid \xi = k \text{ if } y_k = 1,\ k \in I_t;\ \xi = -k, \text{ if } y_k = 0,\ k \in I_t\}.$$

Then, J_t represents a *partial solution* determined at iteration t. A decision term y_k with $k \in \bar I_t = Q\backslash I_t$ is said to be a *free* term of the partial solution J_t. Assigning binary values to all free decision terms of J_t yields a *completion* of J_t. Note that if J_t has l elements, it can determine 2^{q-l} different completions. Among all completions of J_t, the *typical* completion y^t is the completion with all the free y_k's set to be zero. Since all $\tilde{a}_k$'s in the master problem (9) are nonnegative, the typical completion of J_t has the minimum value of the objective function among all completions of J_t.

A partial solution J_t is said to be *feasible* (*infeasible*) if its typical completion constitutes a feasible (*infeasible*) solution y to the master problem (9). A partial solution J_t can be also used to partially determine some decision variables x_j's consistently via the secondary constraints (10) or can lead to an inconsistent solution. When an inconsistency occurs, J_t is said to be an *inconsistent partial solution*. Otherwise, it is a *consistent partial solution*. It is clear an inconsistency of J_t implies that all completions of J_t are inconsistent to the secondary constraints.

5.1.3 Consistency check

When J_t is consistent, the decision variables x_j's determined by the second constraints (10) form the *converted solution* of J_t. The converted solution can be represented by the signed index set:

$$D_t = \{\xi \mid \xi = j \text{ if } x_j = 1,\ j \in d_t;\ \xi = -j,\ \text{if } x_j = 0,\ j \in d_t\},$$

where d_t is the index set of all x_j's in the converted solution. The converted solution D_t could further determine some free decision terms y_k's by the secondary constraints. These determined decision terms constitute an *augmented solution* of J_t which can be represented by the signed index set:

$$B_t = \{\xi \mid \xi = k \text{ if } y_k = 1,\ j \in b_t;\ \xi = -k,\ \text{if } y_k = 0,\ k \in b_t\},$$

where b_t is the index set of all y_k's determined by the converted solution D_t. If B_t is uniquely determined by D_t, then the complement of any element in B_t must lead to an inconsistency and thus all decision terms in the augmented solution can be fixed. We underline a signed index in B_t to denote that this decision term is fixed in the augmented solution. It is clear that a new partial solution $J_{t+1} = J_t \cup B_t$ must be consistent. In the case of $B_t = \emptyset$, J_t itself is consistent.

5.1.4 Computation of feasible partial solutions

Taking the advantage of the single constraint in (9), a simple procedure can be derived to search for a feasible partial solution of (9). Suppose that J_t is a partial solution at iteration t. Let I_t be the index of J_t and y^t the typical completion of J_t. Denote by (MP^t) the master problem (9) with y_k, $k \in I_t$, being fixed at zero or one according to J_t. When $\tilde{g}(y^t) > \tilde{b}$, J_t is an infeasible partial solution. If

$$\tilde{g}(y^t) + \sum_{k \in \bar{I}_t} \min(0, \tilde{a}_k) > \tilde{b} \tag{11}$$

then, it is impossible to augment J_t to obtain a feasible completion. Thus, J_t can be fathomed. Otherwise, there must exist at least one feasible completion of J_t. The following procedure can be used to find a feasible completion of J_t.

Procedure 1 (Search for feasible partial solution)

Given a partial solution J_t and its index set I_t.

Step 0. If (11) holds, exit and there is no feasible completion of J_t. Otherwise, calculate $\alpha = \tilde{g}(y^t)$. Set $I = \{1, \ldots, T\} \backslash I_t$.

Step 1. Calculate $i = \arg\min_{k \in I} \tilde{a}_k$.
Step 2. Set $J_t := J_t \cup \{i\}$. If $\alpha := \alpha + \tilde{a}_i \le \tilde{b}$, exit and J_t is a feasible partial solution. Otherwise, set $I := I \backslash \{i\}$, return to Step 1.

Procedure 1 either finds a feasible partial solution or reports that no feasible completion of J_t can be found. For the details of the generation procedures for converted solution and augmentation solution, readers may refer to Taha [47], as the procedures are basically the same for the singly-constrained master problem in (9) and a general multiply constrained master problem considered in Taha [47].

5.1.5 *Two-level solution scheme for* $(L_\lambda(l))$

Denote $\tilde{g}(y^t)$ by $\tilde{g}^t$ and $\tilde{f}(y^t)$ by $\tilde{f}^t$. Based on the concept of the backtrack scheme [16] and the Taha's method [47], the following two-level solution method can be now proposed for $(L_\lambda(l))$, the Lagrangian relaxation problem with a lower objective level cut. The flow diagram of the algorithm is also given in Fig. 4.

Procedure 2 (Two-level solution method for $(L_\lambda(l))$)

Step 0 Set $J_0 = \emptyset$, $t = 0$, and $f_{\text{opt}} = \infty$.
Step 1 If $\tilde{g}^t \le \tilde{b}$, go to Step 4.
Step 2 If (11) holds, fathom J_t by feasibility and go to Step 9.
Step 3 Apply Procedure 1 to search for a feasible J_t.
Step 4 If $\tilde{f}^t \ge f_{\text{opt}}$, fathom J_t by domination and go to Step 9.
Step 5 Consistency check. If J_t is inconsistent, fathom J_t by consistency and go to Step 9. Otherwise, obtain D_t.
Step 6 B_t recognition. If $B_t = \emptyset$, set $y^* = y^t$ and $f_{\text{opt}} = \tilde{f}^t$, fathom J_t by optimality and go to Step 9. Otherwise, augment J_t with B_t on the right.
Step 7 If $\tilde{f}^t \ge f_{\text{opt}}$, fathom J_t by domination and go to Step 9.
Step 8 If $\tilde{g}^t \le \tilde{b}$, set $y^* = y^t$ and $f_{\text{opt}} = f^t$, fathom J_t by optimality and go to Step 9. Otherwise, set $J_{t+1} = J_t$, $t = t + 1$ and go to Step 2.
Step 9 Backtrack. If all elements in J_t are underlined, terminate the algorithm. Otherwise, generate J_{t+1} by replacing the rightmost element of J_t which is not underlined by its underlined complement and delete all elements to its right. Set $t = t + 1$ and go to Step 1.

5.2 Main algorithm for (P_{0-1})

With Procedure 2 as the solver for $(L_\lambda(l))$, the following convergent Lagrangian and objective level cut algorithm can be now proposed for obtaining an exact solution to (P_{0-1}).

Algorithm 2 (*Convergent Lagrangian and objective level cut method for* (P_{0-1}))

Step 0 (*initialization*) Compute a lower bound l_0 of f^*. Set $t = 0$ and $f_{\text{opt}} = \infty$.
Step 1 If $l_t \ge f_{\text{opt}}$, stop.
Step 2 (*dual search with objective cut*) Solve

$$(D^{l_t}) \quad \max_{\lambda \in \mathbb{R}^m_+} d(\lambda, l_t)$$

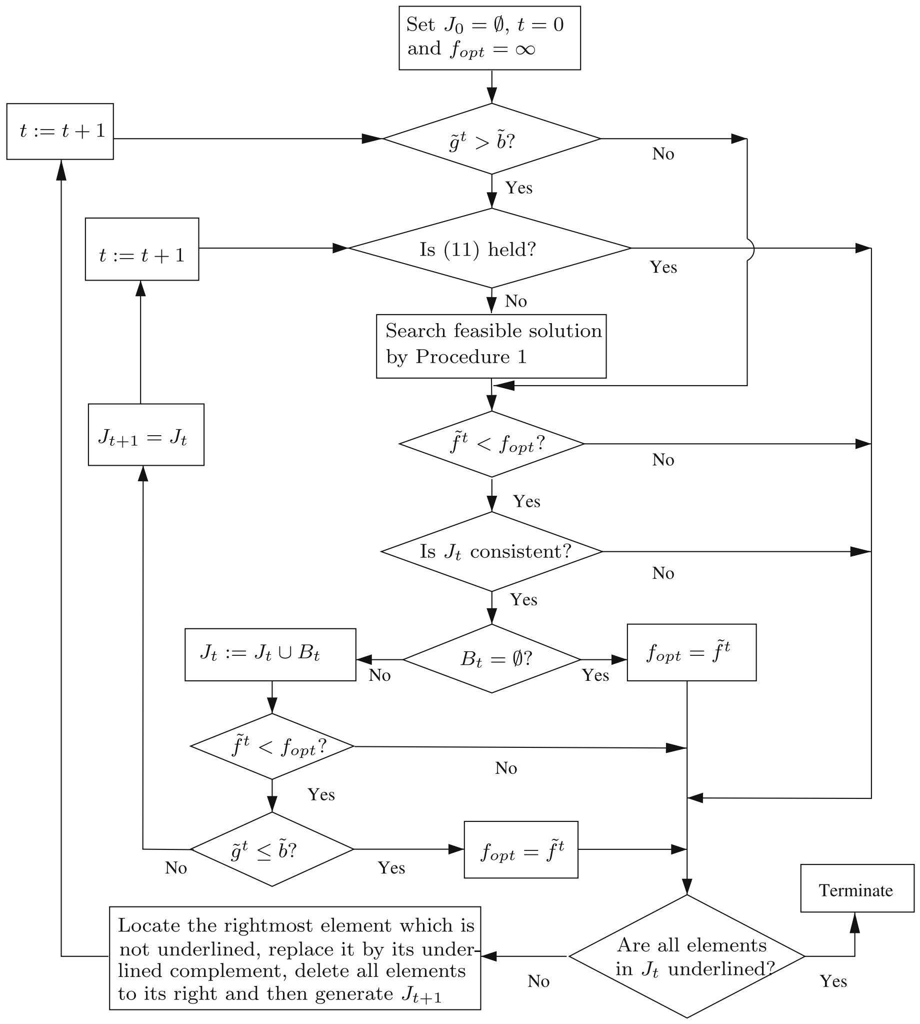
Set $J_0 = \emptyset$, $t = 0$ and $f_{opt} = \infty$
$t := t + 1$
$\tilde{g}^t > \tilde{b}$?
No
Yes
$t := t + 1$
Is (11) held?
Yes
No
Search feasible solution by Procedure 1
$J_{t+1} = J_t$
$\tilde{f}^t < f_{opt}$?
No
Yes
Is J_t consistent?
No
Yes
$J_t := J_t \cup B_t$
$B_t = \emptyset$?
No
Yes
$f_{opt} = \tilde{f}^t$
$\tilde{f}^t < f_{opt}$?
No
Yes
$\tilde{g}^t \leq \tilde{b}$?
No
Yes
$f_{opt} = \tilde{f}^t$
Terminate
Locate the rightmost element which is not underlined, replace it by its underlined complement, delete all elements to its right and then generate J_{t+1}
No
Are all elements in J_t underlined?
Yes

Fig. 4 Flow diagram of the two-level solution scheme for $(L_\lambda(l))$

by some dual search procedure, while the Lagrangian relaxation problem $(L_\lambda(l_t))$ is solved by using Procedure 2. The dual search method terminates when the algorithm is not able to increase the dual value after a given number of iterations. Let λ^t be the dual vector that generates the highest dual value in the dual search process. Set $d^t = d(\lambda^t, l_t)$.

Step 3 If $d^t > l_t$, set $l_{t+1} = \lceil d^t \rceil$ and let $t := t+1$. If a feasible solution $\tilde{x}$ with $f(\tilde{x}) < f_{\text{opt}}$ is found during the dual search process, set $x_{\text{opt}} = \tilde{x}$, set $f_{\text{opt}} = f(\tilde{x})$. Go to Step 1.

Step 4 If $d^t = l_t$, solve the following problem using Procedure 2 with $\lambda = 0$:

$$\begin{aligned} \min\ & f(x) = \sum_{k=1}^{q} c_j \prod_{j\in Q_k} x_j, \\ \text{s.t. }\ & l_t \le f(x) = \sum_{k=1}^{q} c_k \prod_{j\in Q_k} x_j, \\ & x \in \{0,1\}^n. \end{aligned} \tag{12}$$

If there is a feasible optimal solution x^t to (12), stop and x^t is the optimal solution to (P_{0-1}). Otherwise, set $l_{k+1} = f(x^t) + 1$, where x^t is an optimal solution to (12). Set $t := t + 1$ and go to Step 1.

The algorithm enters Step 4 only when the algorithm is not able to raise the dual value at Step 3. Step 4 corresponds to the Lagrangian relaxation problem with $\lambda = 0$. When Step 4 identifies a feasible solution, it will be optimal to the primal problem. When Step 4 is not able to find feasible solutions, it can still help to raise the lower objective cut. The following theorem is obvious and its proof is omitted due to the similarity to the one for Theorem 2.

Theorem 3 *Algorithm 2 either finds an optimal solution of* (P_{0-1}) *or reports an infeasibility of* (P_{0-1}) *in finite number of iterations.*

Now we apply Algorithm 2 to solve the following example:

Example 3

$$\begin{aligned} \min\ & 3x_1 + 5x_1x_2x_3 + 3x_1x_4x_5 + 8x_2x_3x_5 - 4x_3x_4x_5, \\ \text{s.t. }\ & 3x_1 - x_1x_4x_5 - x_2x_3x_5 + x_3x_4x_5 \le 2, \\ & 2x_1 - 4x_1x_2x_3 - 7x_1x_4x_5 - 3x_2x_3x_5 - x_3x_4x_5 \le -3, \\ & -6x_1 - 3x_1x_2x_3 + 5x_1x_4x_5 - 3x_2x_3x_5 + 6x_3x_4x_5 \le 5, \\ & x_1, x_2, x_3, x_4, x_5 \in \{0,1\}. \end{aligned}$$

We set, as initial values, $l^0 = -4$, the incumbent $x_{\text{opt}} = \emptyset$ and $f_{\text{opt}} = \infty$. Table 2 provides the details of the iterative process in obtaining the optimal solution $(0,1,1,1,1)^T$ with an optimal value of 4 in three iterations.

Table 2 Iteration process of Example 3

Iteration	λ^k	d^k	x^*	$f(x^*)$	l_k
0			$\emptyset$	∞	
1	$(0, 1.225, 0.612)^T$	-1.67	$(1,1,1,1,1)^T$	15	-1
2	$(0.264, 0, 0)^T$	3.26	$(1,0,0,1,1)^T$	6	4
3	$(0,0,0)^T$	4	$(0,1,1,1,1)^T$	4	4

6 Computational experiment

We report in this section the computational results in testing Algorithm 1 for five classes of separable integer programming problems and Algorithm 2 for the polynomial zero-one programming problem.

6.1 Test problems

The six classes of test problems are described as follows.

Problem 1 Third degree polynomial integer programming:

$$f_j(x_j) = \sum_{k=1}^{3} c_{jk} x_j^k, \quad j = 1, \ldots, n,$$
$$g_{ij}(x_j) = \sum_{k=1}^{3} a_{ijk} x_j^k, \quad i = 1, \ldots, m, \; j = 1, \ldots, n.$$

Coefficients c_{ik} are integer numbers with $c_{i1} \in [-20, 20]$, $c_{i2} \in [-10, 10]$, and $c_{i3} \in [-5, 5]$. Coefficients a_{ijk} are of real values with $a_{ij1} \in [-20, 20]$, $a_{ij2} \in [-10, 10]$, and $a_{ij3} \in [-5, 5]$.

Problem 2 Convex quadratic integer programming with convex quadratic constraints:

$$f_j(x_j) = c_{j1} x_j^2 + c_{j2} x_j, \quad j = 1, \ldots, n,$$
$$g_{ij}(x_j) = a_{ij1} x_j^2 + a_{ij2} x_j, \quad i = 1, \ldots, m, \; j = 1, \ldots, n.$$

Coefficients c_{j1} and c_{j2}, $j = 1, \ldots, n$, are integer numbers taken from [1,10] and $[-100, 20]$, respectively. Coefficients a_{ij1} and a_{ij2}, $i = 1, \ldots, m, j = 1, \ldots, n$, are of real values taken from $[1, 10]$ and $[100, 220]$, respectively.

Problem 3 Convex quadratic integer programming with linear constraints:

$$f_j(x_j) = c_{j1} x_j^2 + c_{j2} x_j, \quad j = 1, \ldots, n,$$
$$g_{ij}(x_j) = a_{ij} x_j, \quad i = 1, \ldots, m, \; j = 1, \ldots, n.$$

Coefficients c_{j1} and c_{j2}, $j = 1, \ldots, n$, are integer numbers taken from [1,10] and $[-100, 20]$, respectively. Coefficient a_{ij}, $i = 1, \ldots, m, j = 1, \ldots, n$, are of real values taken from $[20, 60]$.

Problem 4 Concave quadratic integer programming with convex quadratic constraints:

$$f_j(x_j) = c_{j1} x_j^2 + c_{j2} x_j, \quad j = 1, \ldots, n,$$
$$g_{ij}(x_j) = a_{ij1} x_j^2 + a_{ij2} x_j, \quad i = 1, \ldots, m, \; j = 1, \ldots, n.$$

Coefficients c_{j1} and c_{j2}, $j = 1, \ldots, n$, are integer numbers taken from $[-10, -1]$ and $[-20, 60]$, respectively. Coefficients a_{ij1} and a_{ij2}, $i = 1, \ldots, m, j = 1, \ldots, n$, are of real values taken from $[1, 10]$ and $[100, 220]$, respectively.

Problem 5 Concave quadratic integer programming with linear constraints:

$$f_j(x_j) = c_{j1} x_j^2 + c_{j2} x_j, \quad j = 1, \ldots, n,$$
$$g_{ij}(x_j) = a_{ij} x_j, \quad i = 1, \ldots, m, \; j = 1, \ldots, n.$$

Coefficients c_{j1} and c_{j2}, $j = 1, \dots, n$, are integer numbers taken from $[-10, -1]$ and $[-20, 60]$, respectively. Coefficient a_{ij}, $i = 1, \dots, m$, $j = 1, \dots, n$, are of real values taken from $[20, 80]$.

All the coefficients in the above five problems are taken uniformly and independently. The finite integer sets are of the following form:

$$X_j = \{x_j \in \mathbb{Z} \mid 1 \le x_j \le 5\}, \quad j = 1, \dots, n.$$

The right-hand side b in the above five problems is generated according to the following rule. Let $0 < r < 1$. Set

$$b_i = \underline{g}_i + r(\bar{g}_i - \underline{g}_i), \quad i = 1, \dots, m,$$

where $\bar{g}_i = \max_{x \in X} g_i(x)$ and $\underline{g}_i = \min_{x \in X} g_i(x)$. The ratio r is used to control the size of the feasible regions of the test problems and the degree of difficulty of the problems. As we will see in the numerical results, the smaller the value of r, the more difficult is the problem. A similar rule of determining the right-hand side was used in generating test problems in Bretthauer and Shetty [7,8].

Problem 6 Polynomial zero-one programming problem (P_{0-1}). The test problems for (P_{0-1}) are randomly generated using the following ranges of the coefficients: $c_k \in [-10, 20]$, $a_{ik} \in [-5, 15]$ and the right-hand side is taken as $b_i = (1-r)\sum_{k=1}^{q} \min(0, a_{ik}) + r\sum_{k=1}^{q} \max(0, a_{ik})$ where $r \in (0, 1)$ is an adjustable ratio of the right-hand side. A density number $D \in (0, 1]$ is also adjustable in controlling the percentage of nonzero coefficients in matrix $A = (a_{ik})_{m \times q}$. The indices in Q_k are randomly generated from set $\{1, \dots, n\}$ with $2 \le |Q_k| \le 6$ for $k = 1, \dots, q$.

6.2 Numerical results

Both Algorithms 1 and 2 have been coded by Fortran 90 and run on a Sun Blade 2000 workstation. The computational results of Algorithm 1 for Problems 1–5 are reported in Tables 3–7, while the computational results of Algorithm 2 for Problem 6 are summarized in Table 8. All the results are obtained by running the algorithm for

Table 3 Numerical results for third degree polynomial integer programming ($r = 0.67$)

n	m	Average duality bounds	Average number of iterations	Average CPU seconds
50	10	430.9	1	2.8
50	20	2,157.1	3	0.7
50	30	1,111.1	4	33.4
50	50	2,802.0	7	58.8

Table 4 Numerical results for convex quadratic integer programming with convex quadratic constraints ($r = 0.62$)

n	m	Average duality bounds	Average number of iterations	Average CPU seconds
50	10	773.5	5	5.3
50	20	795.7	13	334.7
50	25	1,007.7	7	126.7
50	30	986.1	8	194.3

Table 5 Numerical results for convex quadratic integer programming with linear constraints ($r = 0.65$)

n	m	Average duality bounds	Average number of iterations	Average CPU seconds
50	10	5.7	3	113.1
50	15	84.4	6	51.1
50	20	29.4	2	2.5
50	30	18.9	3	381.3

Table 6 Numerical results for concave quadratic integer programming with convex quadratic constraints ($r = 0.70$)

n	m	Average duality bounds	Average number of iterations	Average CPU seconds
50	5	533.5	3	47.4
50	8	765.6	4	5.5
50	10	791.4	6	8.8
50	20	1,624.6	10	266.3

Table 7 Numerical results for concave quadratic integer programming with linear constraints ($r = 0.70$)

n	m	Average duality bounds	Average number of iterations	Average CPU seconds
50	5	70.2	2	0.1
50	10	57.9	6	86.1
50	15	122.0	7	26.8
50	20	132.6	5	1,027.2

20 randomly generated problems. The average number of iterations is rounded off to its nearest integer. The following notations are used in the tables:

- n = number of variables;
- m = number of constraints;
- q = number of decision terms in (P_{0-1});
- r = ratio corresponding to the adjustable right-hand side b;
- D = density of matrix $A = (a_{ik})_{m\times q}$ in (P_{0-1});
- Duality bound = initial duality bound $u_0 - l_0$, where l_0 and u_0 are defined in Algorithm 1.

Tables 3–8 indicate that the proposed algorithms are capable of solving medium-size separable integer programming problems and polynomial 0-1 programming problems within reasonable CPU time. From Tables 3–7, we observe that for a fixed number of variables, the CPU time of Algorithm 1 for solving separable integer programming problems tends to increase as the number of constraints m goes up. This could be partially due to that the duality bound of the problem has the tendency to increase as m increases and that the quality of the best dual value found by the subgradient method becomes poorer as the number of dual variables, m, increases. Table 8 indicates that for a fixed number of nonlinear terms in polynomial 0-1 programming problems, Algorithm 2 becomes more efficient when the number of variables goes up. This could be due to the fact that the back-track scheme for fathoming partial solutions by feasibility and consistency check in the two-level solution method for $(L_\lambda(l))$ becomes more efficient when there are less overlapping variables among Q_k $(k = 1,\ldots,q)$. We can also conclude that the number of nonlinear terms q has a significant impact on

Table 8 Numerical results for 0-1 polynomial programming problem ($r = 0.5$)

q	n	m	Average CPU time (s)			
			$D = 0.25$	$D = 0.50$	$D = 0.75$	$D = 1.0$
50	100	20	1.2	1.0	23.4	8.3
50	150	20	0.4	0.3	13.6	0.6
50	200	20	1.5	0.5	2.5	1.1
70	100	20	70.5	281.2	495.8	371.1
70	150	20	11.0	97.4	134.0	45.2
70	200	20	11.6	37.4	113.8	72.7

Table 9 Comparison results of Algorithm 1 and BARON for third degree polynomial integer programming problem

n	m	Algorithm 1			BARON		
		Average CPU time (s)			Average CPU time (s)		
		$r = 0.63$	$r = 0.65$	$r = 0.67$	$r = 0.63$	$r = 0.65$	$r = 0.67$
30	10	1.0	2.2	0.8	4.4	13.4	9.0
30	20	16.8	15.9	0.3	21.2	22.6	3.9
30	30	18.2	13.8	2.0	20.9	7.7	8.2
40	10	177.4	0.84	0.4	7.7	8.6	10.5
40	20	12.4	14.9	2.0	26.7	43.2	8.0
40	30	57.6	8.1	1.9	48.3	65.1	12.2

the performance of Algorithm 2. In fact, the two-level solution method for solving the Lagrangian relaxation $(L_\lambda(l))$ becomes less efficient as the number of the secondary variables introduced in the two-level reformulation increases.

6.3 Comparison results

We have compared Algorithms 1 and 2 with BARON (Version 7.5.3), a commercial software that can solve mixed-integer nonlinear programming problems to global optimality (see [39]). BARON is based on a prototypical branch-and-bound algorithm that incorporates novel relaxations schemes, range reduction tests, and branching strategies (see [49]). The comparison testing was performed on a Pentium IV PC (2.2 GHz and 256 Mb RAM). Problems 1 and 6 are used in our numerical comparison with BARON. The test problems are first generated and solved by Algorithms 1 and 2 and then solved by BARON using the optimization modeling software AIMMS 3.6 (see [36]).

Tables 9 and 10 summarize the comparison results in which the average CPU time of each instance is obtained by solving 5 randomly generated test problems. We note from Table 9 that Algorithm 1 uses less CPU time to obtain the optimal solution of the third degree polynomial integer programming problems than BARON in most cases. Comparing the results in Table 10, it appears that Algorithm 2 outperforms BARON for 0-1 polynomial problems with $n \geq 150$ while BARON performs better than Algorithm 2 for problems with $n \leq 100$. It is interesting to observe from Table 10 that for a fixed number of nonlinear terms in (P_{0-1}), the CPU time of BARON increases significantly as n increases while the CPU time of Algorithm 2 tends to decrease as n increases.

Table 10 Comparison results of Algorithm 2 and BARON for 0-1 polynomial programming problem ($m = 20, q = 50, r = 0.5$)

n	Algorithm 2				BARON			
	Average CPU time (s)				Average CPU time (s)			
	D = 0.25	0.50	0.75	1.0	D = 0.25	0.50	0.75	1.0
50	7.4	11.5	59.6	4.9	2.6	0.89	4.0	1.3
100	4.6	3.2	17.2	11.9	4.5	2.7	4.2	5.3
150	0.60	2.9	10.3	1.8	82.2	15.2	16.1	5.5
200	0.83	1.4	2.3	2.7	394.8	59.7	NS	NS

NS—5 test problems were not solved within 2 CPU hours

It is worth pointing out BARON is applicable to solve general nonconvex nonseparable MINLP where only factorable functions are involved, while Algorithm 1 is a specific algorithm designed for separable integer programming problem (P_s) which does not require the factorability of the functions involved.

7 Conclusions

A novel Lagrangian dual and objective level cut method has been proposed in this paper. The method is based on a key observation that the duality gap of an integer program can be eliminated by reshaping (re-confining) the perturbation function. Consequently, the optimal solution can be exposed to the convex hull of the revised perturbation function and thus the success of dual search can be guaranteed. We have investigated how to add objective level cuts to reshape the perturbation function. Two specific algorithms have been developed for separable integer programming and polynomial 0-1 programming, respectively. Implementation issues in solving the Lagrangian relaxation problems with an objective cut constraint for both (P_s) and (P_{0-1}) have been also discussed. The numerical results show that the proposed convergent Lagrangian and objective level cut methods are capable of solving medium-size separable integer programming problems and polynomial 0-1 programming problems in reasonable time. Comparison results also show the effectiveness of our algorithms.

Acknowledgements This work was supported by the Research Grants Council of Hong Kong grants CUHK 4214/01E and CUHK4245/04E, and National Natural Science Foundation of China grants 70671064 and 70518001.

References

1. Balas, E.: An additive algorithm for solving linear programs with zero-one variables. Oper. Res. **13**, 517–546 (1965)
2. Balas, E., Mazzola, J.B.: Nonlinear 0-1 programming: I. Linearization Techniques. Math. Program. **30**, 1–21 (1984a)
3. Balas, E., Mazzola, J.B.: Nonlinear 0–1 programming: II. Dominance Relations and Algorithms. Math. Program. **30**, 22–45 (1984b)
4. Bell, D.E., Shapiro, J.F.: A convergent duality theory for integer programming. Oper. Res. **25**, 419–434 (1977)
5. Benson, H.P., Erenguc, S.S.: An algorithm for concave integer minimization over a polyhedron. Nav. Res. Logist. **37**, 515–525 (1990)

6. Bretthauer, K.M., Cabot, A.V., Venkataramanan, M.A.: An algorithm and new penalties for concave integer minimization over a polyhedron. Nav. Res. Logist. **41**, 435–454 (1994)
7. Bretthauer, K.M., Shetty, B.: The nonlinear resource allocation problem. Oper. Res. **43**, 670–683 (1995)
8. Bretthauer, K.M., Shetty, B.: A pegging algorithm for the nonlinear resource allocation problem. Comput. Oper. Res. **29**, 505–527 (2002)
9. Chern, M.S.: On the computational complexity of reliability redundancy allocation in a series system. Oper. Res. Lett. **11**, 309–315 (1992)
10. Dantzig, G.B.: On the significance of solving linear programming with some integer variables. Econometrica **28**, 30–40 (1960)
11. Fisher, M.L.: The Lagrangian relaxation method for solving integer programming problems. Manage. Sci. **27**, 1–18 (1981)
12. Fisher, M.L., Shapiro, J.F.: Constructive duality in integer programming. SIAM J. Appl. Math. **27**, 31–52 (1974)
13. Floudas, C.A.: Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford (1995)
14. Fortet, R.: L′algèbre de Boole et ses applications en recherche Opérationnelle. Cah. Centre d' Étud. Rech. Opér. **1**, 5–36 (1959)
15. Fortet R.: Applications de l′algèbre de Boole en recherche Opérationnelle. Rev. Fr. d' Inform. Rech. Opér. **4**, 17–26 (1960)
16. Geoffrion, A.M.: Integer programming by implicit enumeration and Balas' method. SIAM Rev. **9**, 178–190 (1967)
17. Geoffrion, A.M.: Lagrangean relaxation for integer programming. Math. Program. Study **2**, 82–114 (1974)
18. Granot, F., Hammer, P.L.: On the role of generalized covering problems. Cah. Centre d' Étud. Rech. Opèr. **17**, 277–289 (1975)
19. Grossmann I.E., Kravanja Z.: Mixed-integer nonlinear programming: a survey of algorithms and applications. In: Biegler, L.T., Coleman, T.F., Conn, A.R. and F. N. Santosa (eds.) Large-Scale optimization with Applications, Part II: Optimization Design and Control. Springer, Berlin, Heidelberg, New york (1997)
20. Hansen, P.: Methods of nonlinear 0-1 programming. Ann. Discrete Math. **5**, 53–70 (1979)
21. Hansen, P., Jaumard, B., Mathon, V.: Constrained nonlinear 0-1 programming. ORSA J. Comput. **5**, 97–119 (1993)
22. Helmberg, C., Rendl, F.: Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. Math. Program. **82**, 291–315 (1998)
23. Hochbaum, D.S.: A nonlinear Knapsack problem. Oper. Res. Lett. **17**, 103–110 (1995)
24. Horst, R., Thoai, N.V.: An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks. OR Spektrum **20**, 47–53 (1998)
25. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin, Heidelberg, New york (1993)
26. Ibaraki, T., Katoh, N.: Resource Allocation Problems: Algorithmic Approaches. MIT Press, Cambridge, MA (1988)
27. Lemaréchal, C., Renaud, A.: A geometric study of duality gaps, with applications. Math. Program., **90**, 399–427 (2001)
28. Li, D., Sun, X.L.: Success guarantee of dual search in integer programming: p-th power Lagrangian method. J. Glob. Optim. **18**, 235–254 (2000)
29. Li, D., Sun, X.L.: Towards strong duality in integer programming. J. Glob. Optim. **35**, 255–282 (2006)
30. Li, D., White, D.J.: P-th power Lagrangian method for integer programming. Ann. Oper. Res. **98**, 151–170 (2000)
31. Marsten, R.E., Morin, T.L.: A hybrid approach to discrete mathematical programming. Math. Program. **14**, 21–40 (1978)
32. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I convex underestimating problems. Math. Program. **10**, 147–175 (1975)
33. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
34. Parker, R.G., Rardin, R.L.: Discrete Optimization. Academic, Boston (1988)
35. Poljak, S., Rendl, F., Wolkoswicz, H.: A recipe for semidefinite relaxation for 0-1 quadratic programming. J. Glob. Optim. **7**, 51–73 (1995)

36. Roelofs, M., Bisschop, J.: AIMMS user's Guide. Paragon Decision Technology. http://www.aimms.com/aimms/download/manuals/AIMMS_ 3UG.pdf (2006)
37. Ryoo, H.S., Sahinidis, N.V.: Analysis of bounds for multilinear functions. J. Glob. Optim. **19**, 403–424 (2001)
38. Ryoo, H.S., Sahinidis, N.V.: Global optimization of multilinear problems. J. Glob. Optim. **26**, 387–418 (2003)
39. Sahinidis, N.V.: BARON: branch and reduce optimization navigator, user's manual ver. 4.0. Department of Chemical Engineering, University of Illinois at Urbana Champaign. http://archimedes.scs.uiuc.edu/baron/manuse.pdf (2000)
40. Shapiro, J.F.: A survey of lagrangian techniques for discrete optimization. Ann. Discrete Math. **5**, 113–138 (1979)
41. Sherali, H.D.: Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. Acta Math. Vietnam **22**, 245–270 (1997)
42. Sherali, H.D.: Global optimization of nonconvex polynomial programming problems having rational exponents. J. Glob. Optim. **12**, 267–283 (1998)
43. Sherali, H.D., Wang, H.: Global optimization of nonconvex factorable programming problems. Math. Program. **89**, 459–478 (2001)
44. Smith, E.M., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Comput. Chem. Eng. **21**, S791–S796 (1997)
45. Sun, X.L., Li, D.: Asymptotic strong duality for bounded integer programming: a logarithmic-exponential dual formulation. Math. Oper. Res. **25**, 625–644 (2000)
46. Sun, X.L., Li, D.: Optimality condition and branch and bound algorithm for constrained redundancy optimization in series systems. Optim. Eng. **3**, 53–65 (2002)
47. Taha, H.A.: A Balasian-based algorithm for zero-one polynomial programming. Manage. Sci. **18**, B328–B343 (1972)
48. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer Academic Publishers, Dordrecht (2002)
49. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Program. **99**, 563–591 (2004)
50. Watters, L.J.: Reduction of integer polynomial programming problems to zero-one linear programming problems. Oper. Res. **15**, 1171–1174 (1967)