

LAWLER AND BELL'S LEXICOGRAPHIC ENUMERATION

ALGORITHM FOR POLYNOMIAL BINARY

PROGRAMMING

JOHN WALKER

Nanyang Business School, Nanyang Technological University, Singapore 639798

Abstract

Lawler and Bell (1966) proposed a simple and elegant lexicographic algorithm, subsequently modified by Mao and Wallingford (1968), for solving discrete optimisation problems formulated as polynomial binary programming problems. In this paper the simplicity and elegance of the algorithm are retained when augmented with appropriate data structures for avoiding the replication of function evaluations and a facility for allowing greater flexibility in the fathoming of the binary solution vectors. The use of the augmented algorithm for solving quadratic binary programming problems is examined. Problems of this type have applications in finance, capital budgeting and project selection. Computational results with project selection problems, in which the returns and resource utilisations are interdependent, demonstrate that the modifications significantly reduce computation time.

KEYWORDS: LEXICOGRAPHIC ENUMERATION; POLYNOMIAL BINARY PROGRAMMING; CAPITAL BUDGETING; PROJECT SELECTION

1. Introduction

Consider the polynomial binary programming problem (PBP).

$$\text{PBP: } b^*_0 = \inf g_{10}(w) - g_{20}(w)$$

subject to

$$g_{1i}(w) - g_{2i}(w) \leq b_i, 1 \leq i \leq m,$$

$$w \in B^n,$$

where the polynomial functions $g_{1i}, g_{2i}, 0 \leq i \leq m$, are monotone non-increasing in each of the binary variables $w_j, 1 \leq j \leq n$.

Lawler and Bell (1966) developed a simple and elegant algorithm for solving instances of PBP in which $g_{20}(w)$ is identically equal to zero. Mao and Wallingford (1968, 1969) modified the basic algorithm to allow g_{20} to be non-zero. Henceforth, in this paper the term *basic algorithm* will refer to Mao and Wallingford's modification of Lawler and Bell's original algorithm. This basic algorithm will be labelled LB.

LB is a modification of total enumeration of the 2^n binary solution vectors in *lexicographically increasing order*. In the process of enumerating these vectors, a fathoming test is used on *comparable* vectors to exclude as many non-optimal vectors as possible. In section 2 LB is described and illustrated with a small numerical example. In section 3 the simplicity and elegance of LB are retained when augmented with appropriate data structures for avoiding the replication of function evaluations and a facility for allowing greater flexibility in fathoming binary solution vectors. This augmented algorithm will be labelled LB^+ . In section 4 the use of LB^+ for solving quadratic binary programming problems is examined. Problems of this type have applications in finance, capital budgeting and project selection, see for example,

Carraway and Schmidt (1991) and the references contained therein. Computational results for project selection problems, obtained on a microcomputer, demonstrate that the modifications significantly reduce computation time.

2. The Basic Algorithm LB

Comparable Binary Vectors And Lexicographic Ordering

Two binary vectors w, x are *comparable*, written $w \prec x$, if and only if $w_j \leq x_j$, $1 \leq j \leq n$. In the algorithm for solving PBP, the 2^n binary solution vectors are enumerated according to the following fixed *lexicographic ordering*.

$$L(w) = 2^{n-1}w_1 + 2^{n-2}w_2 + \dots + 2^1w_{n-1} + 2^0w_n$$

LB is a modification of total enumeration in lexicographically increasing order. The enumeration starts with $w = (0, 0, \dots, 0)$ having lowest lexicographic ordering $L(w) = 0$ and terminates when $w = (1, 1, \dots, 1)$ having the highest lexicographic ordering $L(w) = 2^n - 1$. In the process of enumerating these vectors, a fathoming test is used on comparable vectors to exclude as many non-optimal vectors as possible.

Fathoming Binary Vectors

If the vectors are ordered by L , fathoming is permissible when it can be shown that every vector x such that $w \prec x \prec y^0$ and $L(w) \leq L(x) \leq L(y^0)$ is not optimal in PBP and the next vector considered for enumeration is $z^0 \in B^n$ with $L(z^0) = L(y^0) + 1 > L(w) + 1$ (i.e., treating z^0 and y^0 as binary numbers $z^0 = y^0 + 1$). Non successful fathoming means that the next vector considered for enumeration is $w = w + 1$ where the updated w may be constructed by a call to a procedure *update_w*.

Construction of Comparable Binary Vectors

In order to implement the fathoming test it is necessary, for an arbitrary vector w , to be able to construct the binary vector y^0 such that $w \prec x \prec y^0$ and $L(w) \leq L(x) \leq L(y^0)$. To this end, define index s to be the position containing the *rightmost one* in w ($s = 0$ if no such rightmost one exists), and index r to be the position containing the *rightmost zero to the left* of s ($r = 0$ if no such rightmost zero exists). It is clear that w can be one of the three forms.

Form 1. $s = 0$ with $w = (0, \dots, 0)$ and the construction of y^0 is given by $y^0 = (1, \dots, 1)$.

Form 2. $1 \leq s \leq n$, $r = 0$ with $w = (1, \dots, 1_s, 0_{s+1}, \dots, 0)$ and the construction of y^0 is given by $y^0 = (1, \dots, 1_s, 1_{s+1}, \dots, 1)$.

Form 3. $1 \leq s \leq n$, $1 \leq r \leq s-1$, with $w = (w_1, \dots, w_{r-1}, 0_r, 1_{r+1}, \dots, 1_s, 0_{s+1}, \dots, 0)$ and the construction of y^0 is given in the following manner.

Define

$$C(w) = \{x: w \prec x, x \in B^n\},$$

$$N(w) = \{x: L(w) < L(x), x \in B^n\}, \text{ and}$$

$$L(z^0) = \min \{L(x): x \in N(w)-C(w)\}.$$

Thus z^0 is defined to be the first vector succeeding w in the (N)umerical, lexicographic, ordering such that w is not (C)omparable with z^0 .

Proposition 1. z^0 is given by the following construction

$$z^0 = (w_1, \dots, w_{r-1}, 1_r, 0_{r+1}, \dots, 0_s, 0_{s+1}, \dots, 0).$$

The proof of construction is given by Garfinkel and Nemhauser (1972, pp. 346). Define $y^0 \in B^n$ to be the binary vector satisfying $L(y^0) = L(z^0)-1$, i.e., $y^0 = z^0-1$. Then, y^0 has the construction

$$y^0 = (w_1, \dots, w_{r-1}, 0_r, 1_{r+1}, \dots, 1_s, 1_{s+1}, \dots, 1).$$

All three forms of w, y^0 may be constructed by a call to a procedure *construct_y*.

Define

$$F^0(w) = \{x: x \in B^n, L(w) \leq L(x) \leq L(y^0)\}.$$

Then, by the construction of y^0 , $w \prec x \prec y^0$ for all $x \in F^0(w)$. The following proposition allows for fathoming of vectors in the enumeration process.

Proposition 2 If g_{1i}, g_{2i} are monotonic non-increasing functions, then $g_{1i}(w) \geq g_{1i}(x) \geq g_{1i}(y^0)$ and $g_{2i}(w) \geq g_{2i}(x) \geq g_{2i}(y^0)$ for all $x \in F^0(w)$.

Proof By the definition of y^t , $w \prec x \prec y^0$ for all $x \in F^0(w)$. Then, from the monotonicity of g_{1i}, g_{2i} , the proposition follows.

Fathoming of vectors is then justified by proposition 2. Consider a constraint i and the vectors w and y^0 with $g_{1i}(y^0)-g_{2i}(w) > b_i$. For $x \in F^0(w)$, $g_{1i}(w) \geq g_{1i}(x) \geq g_{1i}(y^0)$ and $-g_{2i}(w) \leq -g_{2i}(x) \leq -g_{2i}(y^0)$. Therefore, $g_{1i}(x)-g_{2i}(x) \geq g_{1i}(y^0)-g_{2i}(w)$. Thus, if $g_{1i}(y^0)-g_{2i}(w) > b_i$ then no $x \in F^0(w)$

can satisfy the i th constraint and, therefore, all $x \in F^0(w)$ are fathomed. The objective function can be treated as a regular constraint of the type $g_{10}(w) - g_{20}(w) \leq b_0$ where b_0 is the updatable objective function value necessary to improve upon the incumbent's objective function value.

The above remarks are the basis for LB described below. On input it is assumed that w^* contains the best known feasible solution and b^*_0 the associated objective function value (a vector $(9, 9, \dots, 9)$ and ∞ respectively if no such feasible solution is known. The initialisation of w and b_0 is implemented by a call to a procedure *initialise*.

PROCEDURE *LB* (w^* , b^*_0)

CALL *initialise*

DO WHILE ($L(w) \leq 2^n - 1$)

CALL *construct_y*

fathomed = .FALSE.

DO $i = 0, m$

IF ($g_{1i}(y^0) - g_{2i}(w) > b_i$) THEN

fathomed = .TRUE.

EXIT

END IF

END DO

IF (fathomed) THEN

$w = y^0$

ELSE

infeasible = .FALSE.

DO $i = 0, m$

```

IF (  $g_{1i}(w) - g_{2i}(w) > b_i$  ) THEN
    infeasible = .TRUE.
    EXIT
END IF
END DO
IF ( .NOT. infeasible ) THEN
     $w^* = w, b^*_0 = g_{10}(w) - g_{20}(w), b_0 = b^*_0$ 
END IF
END IF
CALL update_w
END DO
END LB

```

Numerical Example

$$\text{Min } [\quad] - [-2w_1w_3 - 3w_1w_5 - 4w_2 - 2w_4 - 3w_5]$$

subject to

$$[-w_1w_4 - 3w_2w_5] - [-w_1 - 2w_3 - w_4w_5] \leq -1 \quad ,$$

$$[-2w_1 - 5w_3] - [-2w_4 - 3w_5] \leq -2 \quad ,$$

$$w \in B^5 .$$

A trace of the optimal solution when applying LB is summarised in table 1.

TABLE 1
Solution Trace for LB

w	y ⁰	Fathoming Test		Comment	Feasibility Test		Comment	Update Incumbent
		i	g _{1i} (y ⁰) - g _{2i} (w)		i	g _{1i} (w) - g _{2i} (w)		
00000	11111	0	[0] - [0] = 0	Not Fathomed	0	[0] - [0] = 0	Infeasible	
		1	[-4] - [0] = -4		1	[0] - [0] = 0		
		2	[-7] - [0] = -7					
00001	00001	0	[0] - [- 3] = 3	Fathomed				
		1	[0] - [0] = 0					
00010	00011	0	[0] - [- 2] = 2	Fathomed				
		1	[0] - [0] = 0					
00100	00111	0	[0] - [0] = 0	Fathomed				
		1	[0] - [- 2] = 2					
01000	01111	0	[0] - [- 4] = 4	Not Fathomed	0	[0] - [-4] = 4	Infeasible	
		1	[-3] - [0] = -3		1	[0] - [0] = 0		
		2	[-5] - [0] = -5					
01001	01001	0	[0] - [- 7] = 7	Fathomed				
		1	[-3] - [0] = -3					
		2	[0] - [- 3] = 3					
01010	01011	0	[0] - [- 6] = 6	Fathomed				
		1	[-3] - [0] = -3					
		2	[0] - [- 2] = 2					
01100	01111	0	[0] - [- 4] = 4	Not Fathomed	0	[0] - [-4] = 4	Infeasible	
		1	[-3] - [- 2] = -1		1	[0] - [-2] = 2		
		2	[-5] - [0] = -5					
01101	01101	0	[0] - [- 7] = 7	Not Fathomed	0	[0] - [-7] = 7	Feasible	w* = (01101) b* ₀ = [0]-[-7] = 7 b ₀ = 6
		1	[-3] - [- 2] = -1		1	[-3] - [-2] = -1		
		2	[-5] - [- 3] = -2		2	[-5] - [-3] = -2		
01110	01111	0	[0] - [- 6] = 6	Not Fathomed	0	[0] - [-6] = 6	Infeasible	
		1	[-3] - [- 2] = -1		1	[0] - [-2] = 2		
		2	[-5] - [- 2] = -3					
01111	01111	0	[0] - [- 9] = 9	Fathomed				
10000	11111	0	[0] - [0] = 0	Not Fathomed	0	[0] - [0] = 0	Infeasible	
		1	[-4] - [- 1] = -3		1	[0] - [-1] = 1		
		2	[-7] - [0] = -7					
10001	10001	0	[0] - [- 6] = 6	Fathomed				
		1	[0] - [- 1] = 1					
10010	10011	0	[0] - [- 2] = 2	Fathomed				
		1	[-1] - [- 1] = 0					

TABLE 1 (Continued)
Solution Trace for LB

w	y^0	Fathoming Test i $g_{1i}(y^0) - g_{2i}(w)$	Comment	Feasibility Test i $g_{1i}(w) - g_{2i}(w)$	Comment	Update Incumbent
10100	10111	0 [0] - [- 2] = 2				
		1 [-1] - [- 3] = 2	Fathomed			
11000	11111	0 [0] - [- 4] = 4		0 [0] - [-4] = 4		
		1 [-4] - [- 1] = -3		1 [0] - [-1] = 1	Infeasible	
		2 [-7] - [0] = -7	Not Fathomed			
11001	11001	0 [0] - [-10] = 10	Fathomed			
11010	11011	0 [0] - [- 6] = 6				
		1 [-4] - [- 1] = -3				
		2 [-2] - [- 2] = 0	Fathomed			
11100	11111	0 [0] - [- 6] = 6		0 [0] - [-6] = 6		
		1 [-4] - [- 3] = -1		1 [0] - [-3] = 3	Infeasible	
		2 [-7] - [0] = -7	Not Fathomed			
11101	11101	0 [0] - [-12] = 12	Fathomed			
11110	11111	0 [0] - [- 8] = 8	Fathomed			

3 The Modified Algorithm LB⁺

Table 1 indicates that the number of function evaluations is 134. Some of these evaluations are clearly redundant. Thus, if during the fathoming test, the $g_{2i}(w)$ values are stored in an array $g2(i)$ then the array can be used in the feasibility test. The number of function evaluations is then reduced to 116. Further reductions can be obtained in the following manner. A sorted linked list, ordered by the values $L(y)$, of $g1$ function evaluations will be needed. The initialisation of the list structure is implemented in the procedure *initialise*.

PROCEDURE *initialise* (w, s, b_0, b^*_0)

$$w_j = 0, 1 \leq j \leq n; s = 0; b_0 = b^*_0$$

set up linked list structure

END *initialise*

At the first construction of a *particular* y^0 vector, a node is inserted into the sorted linked list data structure ordered by the key $g1(\text{node}, m+1) = L(y^0)$ and with records $g1(\text{node}, i) = \infty, 0 \leq i \leq m$. On subsequent constructions of y^0 its associated node is identified. The determination of (a) the first construction of a particular y^0 vector and subsequent node insertion, or (b) the node identity; is implemented in the procedure *construct_y* with $t = 0$.

PROCEDURE *construct_y* ($w, s, t, y^t, \text{node}$)

$$y^t_j = w_j, 1 \leq j \leq s+t; y^t_j = 1, s+t+1 \leq j \leq n$$

insertion and/or identification of node

END *construct_y*

Subsequently at the fathoming test the node identity is known. If the record $g1(\text{node}, i) = \infty$ the $g_{li}(y^0)$ value is evaluated, the record updated such that $g1(\text{node}, i) = g_{li}(y^0)$ and is made

available for use in the test. If the record $g_1(\text{node}, i) < \infty$ it represents some previous evaluation and is made available for use in the test. At the feasibility test the sorted linked list has to be examined for a node with key $g_1(\text{node}, m+1) = L(w)$. If no such node exists $g_{li}(y^0)$ is evaluated and made available for the test. It is assumed that the detection of the appropriate node and determination of $g_{li}(y^0)$ is implemented in a procedure *determine_g_{li}*. At the completion of the fathoming and feasibility tests all nodes in the list with key $g_1(\text{node}, m+1) \leq L(w)$ are deleted from the list. It is assumed that such deletion is implemented in the procedure *update_w*.

PROCEDURE *update_w* (w, s)

delete all nodes from linked list with key $g_1(\text{node}, m+1) \in L(w)$

s = 0

DO j = n, 1, -1

IF (w_j = 0) THEN

s = j

EXIT

END IF

END DO

IF (s > 0) THEN

w_s = 1; w_j = 0, s+1 ≤ j ≤ n

END IF

END *update_w*

The use of such a data structure reduces the number of function evaluations to 95. Table 1 also indicates that, in some cases where the fathoming test using y^0 fails, it may be possible to

allow for greater flexibility in the fathoming of the binary solution vectors. This may be achieved by considering vectors $w \prec x \prec y^0$ and $L(w) \leq L(x) < L(y^0)$ with a number of variables pegged to their current zero value. For example, the fathoming test with $w = (00000)$ and $y^0 = (11111)$ fails and the enumeration continues with $w = (00001)$. It may be that by pegging $w_1 = 0$ the fathoming test with $w = (00000)$ and $y^1 = (01111)$ succeeds and the enumeration could continue with $w = (10000)$. To this end, for an arbitrary vector w , define indices s and r as in section 2. Define t to be the number of variables *to the immediate right of position s* , $0 \leq t \leq n-s$, which can be pegged to their current zero value for the fathoming test on w . As in section 2 there are three forms of w to consider.

Form 1. $s = 0, 1 \leq t \leq n-s, w = (0_1, \dots, 0_t, 0_{t+1}, \dots, 0)$ and the construction of y^t is given by $y^t = (0_1, \dots, 0_t, 1_{t+1}, \dots, 1)$,

Form 2. $1 \leq s \leq n, r = 0, 1 \leq t \leq n-s, w = (1, \dots, 1_s, 0_{s+1}, \dots, 0_{s+t}, 0_{s+t+1}, \dots, 0)$ and the construction of y^t is given by $y^t = (1, \dots, 1_s, 0_{s+1}, \dots, 0_{s+t}, 1_{s+t+1}, \dots, 1)$.

Form 3. $1 \leq s \leq n, 1 \leq r \leq s-1$ and $1 \leq t \leq n-s$,

$w = (w_1, \dots, w_{r-1}, 0_r, 1_{r+1}, \dots, 1_s, 0_{s+1}, \dots, 0_{s+t}, 0_{s+t+1}, \dots, 0)$ and the construction of y^t is given by $y^t = (w_1, \dots, w_{r-1}, 0_r, 1_{r+1}, \dots, 1_s, 0_{s+1}, \dots, 0_{s+t}, 1_{s+t+1}, \dots, 1)$.

All three forms y^t may be constructed by a call to the procedure *construct_y*.

Define, in a similar manner to $F^0(w)$,

$$F^t(w) = \{x: x \in B^n, L(w) \leq L(x) \leq L(y^t)\}.$$

Then, by the construction of y^t , $w \prec x \prec y^t$ for all $x \in F^t(w)$. Substituting y^t for y^0 and $F^t(w)$ for $F^0(w)$ proposition 2 then again allows for fathoming of vectors in the enumeration process. Similarly, earlier remarks made regarding function evaluations and data structures also carry over.

The pegging of variables to zero allows greater flexibility in fathoming in the sense that if the fathoming test fails at y^t it may be that, after pegging an appropriate variable to zero, the fathoming test succeeds at y^{t+1} . Note that in all cases, $0 \leq t \leq n-s$,

$$y^t = (w_1, \dots, w_{s+t}, 1_{s+t+1}, \dots, 1)$$

and

$$\begin{aligned} L(y^t) - L(w) &= 2^{n-s-t-1} + 2^{n-s-t-2} + \dots + 1 \\ &= 2^{n-s-t} - 1 \end{aligned}$$

The value $L(y^t) - L(w)$ can be used to control the extent of the pegging of variables to zero. Thus, if $L(y^t) - L(w) < \text{peg_min}$, an input control parameter, no further pegging of variables is attempted in the fathoming of the vector w . Note that setting $\text{peg_min} = 2^n$ allows for the option of not pegging variables to zero. The above remarks are the basis for LB^+ described below.

PROCEDURE LB^+ (w^* , b^*_0)

CALL *initialise* (w , s , b_0 , b^*_0)

DO WHILE ($L(w) \leq 2^n - 1$)

DO $t = 0, n-s$

```

CALL construct_y ( w, s, t,  $y^t$ , node )

fathomed = .FALSE.

DO i = 0, m

    CALL determine_gli ( i,  $y^t$ , g1(node,i) )

    IF ( t = 0 ) g2(i) = g2i( w )

    IF ( g1(node,i) - g2(i) > bi ) THEN

        fathomed = .TRUE.

        EXIT

    END IF

END DO

IF ( fathomed ) THEN

    w = y

ELSE

    infeasible = .FALSE.

    DO i = 0, m

        CALL determine_gli ( i, w, g1(node,i) )

        IF ( g1(node,i) - g2(i) > bi ) THEN

            infeasible = .TRUE.

            EXIT

        END IF

    END DO

    IF ( .NOT. infeasible ) THEN

        w* = w, b*0 = g1(node,0) - g2(0), b0 = b*0

    END IF

END IF

CALL update_w ( w, s )

```

```
IF ( L (yt) - L( w ) < peg_min ) EXIT
```

```
END DO
```

```
END DO
```

```
END LB+
```

The results of LB^+ (with the control parameter $peg_min = 4$) on the numerical example of section 2 are summarised in Table 2. The number of function evaluations is reduced to 70.

TABLE 2
Solution Trace for LB⁺ with peg_min = 4

w	t	y ^t	i	Fathoming Test g _{1i} (y ^t) - g _{2i} (w)	Comment	Feasibility Test i g _{1i} (w) - g _{2i} (w)	Comment	Update Incumbent
00000	0	11111	0	[0] - [0] = 0		0 [0] - [0] = 0		
			1	[-4] - [0] = -4		1 [0] - [0] = 0	Infeasible	
			2	[-7] - [0] = -7	Not Fathomed			
	1	01111	0	[0] - [0] = 0		0 [0] - [0] = 0		
			1	[-3] - [0] = -3		1 [0] - [0] = 0	Infeasible	
			2	[-5] - [0] = -5	Not Fathomed			
	2	00111	0	[0] - [0] = 0				
			1	[0] - [0] = 0	Fathomed			
01000	0	01111	0	[0] - [- 4] = 4		0 [0] - [-4] = 4		
			1	[-3] - [0] = -3		1 [0] - [0] = 0	Infeasible	
			2	[-5] - [0] = -5	Not Fathomed			
	1	01011	0	[0] - [- 4] = 4				
			1	[-3] - [0] = -3				
			2	[0] - [0] = 0	Fathomed			
01100	0	01111	0	[0] - [- 4] = 4		0 [0] - [-4] = 4		
			1	[-3] - [- 2] = -1		1 [0] - [-2] = 2	Infeasible	
			2	[-5] - [0] = -5	Not Fathomed			
01101	0	01101	0	[0] - [- 7] = 7		0 [0] - [-7] = 7	Feasible	w* = (01101)
			1	[-3] - [- 2] = -1		1 [-3] - [-2] = -1		b* ₀ = [0]-[-7] = 7
			2	[-5] - [- 3] = -2	Not Fathomed	2 [-5] - [-3] = -2		b ₀ = 6
01110	0	01111	0	[0] - [- 6] = 6		0 [0] - [-6] = 6		
			1	[-3] - [- 2] = -1		1 [0] - [-2] = 2	Infeasible	
			2	[-5] - [- 2] = -3	Not Fathomed			
01111	0	01111	0	[0] - [- 9] = 9	Fathomed			
10000	0	11111	0	[0] - [0] = 0		0 [0] - [0] = 0		
			1	[-4] - [- 1] = -3		1 [0] - [-1] = 1	Infeasible	
			2	[-7] - [0] = -7	Not Fathomed			
	1	10111	0	[0] - [0] = 0				
			1	[-1] - [- 1] = 0	Fathomed			
11000	0	11111	0	[0] - [- 4] = 4		0 [0] - [-4] = 4		
			1	[-4] - [- 1] = -3		1 [0] - [-1] = 1	Infeasible	
			2	[-7] - [0] = -7	Not Fathomed			
	1	11011	0	[0] - [- 4] = 4		0 [0] - [-4] = 4		
			1	[-4] - [- 1] = -3		1 [0] - [-1] = 1	Infeasible	
			2	[-2] - [0] = -2	Not Fathomed			

TABLE 2 (Continued)
Solution Trace for LB⁺ with peg_min = 4

w	t	y ^t	Fathoming Test		Comment	Feasibility Test		Comment	Update Incumbent
			i	$g_{1i}(y^t) - g_{2i}(w)$		i	$g_{1i}(w) - g_{2i}(w)$		
11001	0	11001	0	$[0] - [-10] = 10$	Fathomed				
11010	0	11011	0	$[0] - [-6] = 6$					
			1	$[-4] - [-1] = -3$					
			2	$[-2] - [-2] = 0$	Fathomed				
11100	0	11111	0	$[0] - [-6] = 6$		0	$[0] - [-6] = 6$		
			1	$[-4] - [-3] = -1$		1	$[0] - [-3] = 3$	Infeasible	
			2	$[-7] - [0] = -7$	Not Fathomed				
11101	0	11101	0	$[0] - [-12] = 12$	Fathomed				
11110	0	11111	0	$[0] - [-8] = 8$	Fathomed				

4. Computational Experience

Quadratic Binary Programming

The quadratic binary programming problem has wide application in finance, capital budgeting and scheduling, see for example Carraway and Schmidt (1991) and the references contained therein.

Project Selection

Consider the following project selection problem. A firm has a fixed amount of m resources and a set n projects each of which utilises resources and yields returns. The projects are *interdependent* in that an individual project's return and resource utilisation are influenced by which other projects are selected. It is assumed that it is sufficient to consider only pairwise interactions between projects, and the effect of the interaction is accurately reflected by pairwise-interaction terms that can be added to the sum of the individual returns and resource utilisation of the chosen projects. The problem can then be modelled as the following quadratic binary programming problem (QBP).

$$\text{QBP} \quad \max \sum_{j=1}^n a_{0jj} w_j + \sum_{j=1}^{n-1} \sum_{k=j+1}^n a_{ojk} w_j w_k$$

subject to

$$\sum_{j=1}^n a_{ijj} w_j + \sum_{j=1}^{n-1} \sum_{k=j+1}^n a_{ijk} w_j w_k \leq b_i, \quad 1 \leq i \leq m,$$

$$w \in B^n,$$

where

a_{0jj} is the independent return for project j ,

a_{ojk} is the interactive impact on return when projects j and k are jointly selected,

a_{ijj} is the independent utilisation of resource i for project j ,

a_{ijk} is the interactive impact on utilisation of resource i when projects j and k are jointly selected,

b_i is the availability of resource i , and

w_j is a binary variable equal to one if project j is selected, zero otherwise.

Problem Generation

Test problems were generated according to the following parameters. The values used in the initial computational testing are indicated.

m the number of resource constraints {4, 8 and 12},

n the number of projects {9, 12, and 15},

p_1 the mean of the a_{ij} coefficients, $0 \leq i \leq m$, {100},

p_2 determines the range of the distribution of the a_{ij} coefficients, $0 \leq i \leq m$, {0.1 and 0.5},

p_3 $\Pr[a \text{ given } a_{ijk} \neq 0]$, $0 \leq i \leq m$, $j+1 \leq k \leq n$, {0.1 and 0.9},

p_4 $\Pr[a \text{ given } a_{0jk} > 0 / a_{0jk} \neq 0]$, $j+1 \leq k \leq n$, {0.5}

p_5 $\Pr[a \text{ given } a_{ijk} > 0 / a_{ijk} \neq 0]$, $1 \leq i \leq m$, $j+1 \leq k \leq n$, {0.05}

p_6 determines the magnitude of a_{ijk} , $0 \leq i \leq m$, $j+1 \leq k \leq n$, {0.1 and 0.3},

p_7 determines the magnitude of b_i , $1 \leq i \leq m$, {0.2, 0.3, 0.4, 0.5, 0.6, 0.7 and 0.8}.

The a_{0ij} were generated from a $U[(1-p_2)p_1, (1+p_2)p_1]$ distribution. To generate the return interaction terms, a_{0jk} , it was first determined (using p_3) whether $a_{0jk} \neq 0$ and if so, its sign and magnitude. Given that it was non-zero its sign was determined using p_4 . The magnitude of a_{0jk} was then generated from a $U[0, p_6 a_{0ij}]$ distribution. Hence, problems were generated with either 10% or 90% of the return interactions equal to zero, 50% of the non-zero interactions terms positive, and the maximum value of the return interaction terms either 10% or 30% of the individual project returns. The resource utilisation terms were generated in a similar

fashion, except that the probability of a non-zero interaction being positive, p_5 , was set to 0.05, reflecting the notion that most resource interactions constitute savings. The b_i were generated by setting $b_i = p_7(\sum_{j=1}^n a_{ij})$. Five replications were performed for each combination of problem parameters.

The algorithms LB and LB⁺ were coded in Microsoft Fortran Powerstation Version 1.0 and executed on a Pentium PC clone running at 90 Mhz. All times reported are in seconds. Time for input/output and generation of test data is not included. All comparisons are based on identical data.

Results

Extensive preliminary computational experimentation indicated that the optimal choice for peg_min in LB⁺ was given by the rule that “if $p_7 \leq 0.5$ then $peg_min = 2^n$ else $peg_min = 4$ ”. This rule was implemented for all the results in the following discussion.

TABLE 3
Ratio of Function Evaluations of LB to LB⁺

p_7	m								
	4			8			12		
	n			n			n		
	9	12	15	9	12	15	9	12	15
0.2	0.762	0.806	0.819	0.741	0.791	0.823	0.723	0.782	0.825
0.3	0.711	0.757	0.635	0.715	0.757	0.660	0.705	0.757	0.751
0.4	0.663	0.622	0.614	0.640	0.678	0.601	0.657	0.685	0.593
0.5	0.563	0.555	0.574	0.578	0.542	0.574	0.562	0.553	0.572
0.6	0.409	0.424	0.415	0.449	0.440	0.433	0.451	0.460	0.467
0.7	0.292	0.308	0.309	0.315	0.342	0.360	0.337	0.357	0.383
0.8	0.180	0.202	0.191	0.189	0.214	0.200	0.211	0.232	0.224

For given values of m , n , and p_7 Table 3 compares the performance of LB relative to LB^+ averaged across the remaining parameters p_1, \dots, p_6 . For given values of m and n Table 3 indicates NB^+ 's superior computational performance relative to LB appears to increase dramatically as the amount of available resource (p_7) increases (and hence the number of projects that can be selected increases).

TABLE 4
Computation Time (in Seconds) of LB^+

p_7	m								
	4			8			12		
	n			n			n		
	9	12	15	9	12	15	9	12	15
0.2	0.006	0.072	0.740	0.006	0.075	0.844	0.009	0.080	0.937
0.3	0.009	0.126	0.614	0.014	0.157	0.899	0.015	0.184	1.820
0.4	0.020	0.128	1.011	0.021	0.218	1.303	0.022	0.264	1.527
0.5	0.013	0.142	1.306	0.025	0.184	1.845	0.034	0.247	2.179
0.6	0.011	0.113	0.765	0.025	0.174	1.224	0.033	0.222	1.725
0.7	0.011	0.058	0.376	0.012	0.097	0.651	0.033	0.138	0.857
0.8	0.008	0.030	0.134	0.012	0.049	0.170	0.020	0.073	0.273

For given values of m , n , and p_7 Table 4 indicates the computation time of LB^+ averaged across the 5 replicates and the remaining parameters p_1, \dots, p_6 . For given values of m and n Table 4 seems to indicate that the computation time of LB^+ is a quadratic function of p_7 and achieves a maximum in a region centred around $p_7 = 0.5$. In this region, for fixed n , Table 4 seems to indicate that the computation time of LB^+ is a linear function of m . Finally, although for the given parameters the computation times are trivial, Table 4 clearly indicates the exponential relationship of the computation time of LB^+ with n .

TABLE 5

**Computation Time (in Seconds) of LB^+ with $m = 12$, $n = 21$, $p_1 = 100$, $p_4 = 0.5$, $p_5 = 0.05$
and $p_7 = 0.6$**

		p_2			
		0.10		0.50	
		p_3		p_3	
		0.10	0.90	0.10	0.90
p_6	0.10	325	34	77	15
	0.30	148	45	72	22

It is of some interest to determine in what region of the p_2 , p_3 , p_6 space are problems “harder” or “easier”. To this end the relationship between the computation time and the parameters p_2 (range of distribution of the a_{ij} coefficients), p_3 (probability of a non-zero interactive a_{ijk} coefficient) and p_6 (magnitude of interactive a_{ijk} coefficient) was examined. Table 5 indicates the computation time of LB^+ for $m = 12$, $n = 21$, $p_1 = 100$, $p_4 = 0.5$, $p_5 = 0.05$ and $p_7 = 0.6$. Table 5 indicates the performance of LB^+ improves as the range of distribution of the a_{ij} coefficients and the probability of a non-zero interactive a_{ijk} coefficient increases. The effect of the magnitude of the interactive a_{ijk} coefficients seems minimal except when both the range of distribution of the a_{ij} coefficients and the probability of a non-zero interactive a_{ijk} coefficient are small.

References

CARRAWAY, R.L. AND R.L. SCHMIDT, "An improved discrete dynamic programming algorithm for allocating resources among interdependent projects." *Management Science*, 37 (1991), 1195-1200.

GARFINKEL, R. AND G. NEMHAUSER, *Integer Programming*, Wiley, New York, 1972.

LAWLER, E. L. AND M.D. BELL, "A method for solving discrete optimisation problems." *Operations Research*, 14 (1966), 1098-1112.

MAO, J.C.T. AND B.A. WALLINGFORD, "An extension of Lawler and Bell's method of discrete optimisation with examples from capital budgeting." *Management Science*, 15 (1968), B51-B60.

MAO, J.C.T. AND B.A. WALLINGFORD, "Corrections and Comments." *Management Science*, 16 (1969), 481.