Journal of Industrial Engineering International *Islamic Azad University, South Tehran Branch*

January 2008, Vol. 4, No. 6, 39-50

A generalized implicit enumeration algorithm for a class of integer nonlinear programming problems

M. S. Sabbagh*

Assistant Professor, Dep. of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan, Iran

M. Roshanjooy

M.Sc., Dep. of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan, Iran

Abstract

Presented here is a generalization of the implicit enumeration algorithm that can be applied when the objective function is being maximized and can be rewritten as the difference of two non-decreasing functions. Also developed is a computational algorithm, named linear speedup, to use whatever explicit linear constraints are present to speedup the search for a solution. The method is easy to understand and implement, yet very effective in dealing with many integer programming problems, including knapsack problems, reliability optimization, and spare allocation problems. To see some application of the generalized algorithm, we notice that the branch-and-bound is the popular method to solve integer linear programming problems. But branch-and-bound cannot efficiently solve all integer linear programming problems. For example, De Loera *et al*. in their 2005 paper discuss some knapsack problems that CPLEX cannot solve in hours. We use our generalized algorithm to find a global or near global optimal solutions for those problems, in less than 100 seconds. The algorithm is based on function values only; it does not require continuity or differentiability of the problem functions. This allows its use on problems whose functions cannot be expressed in closed algebraic form. The reliability and efficiency of the proposed algorithm has been demonstrated on some integer optimization problems taken from the literature.

Keywords: Algebraic form; Function values; Generalized implicit enumeration; Integer programming; Linear speedup

1. Introduction

Many practical optimization problems involve decision variables whose values have to be integers. Examples of such problems abound in applications; for example plant operation, design, location, scheduling, set covering, set partitioning, set packing problems and allocation models, etc.

The earlier methods for solving Integer Programming (IP) problems are due to Land and Doig [10] and to Gomory [7]. These methods are designed to solve limited subclasses of IP problems rather than the general category of IP problems. In some instances, the general IP problems can be approximated and solved as a Linear Programming (LP) problem and then the solution to that LP optimization are rounded. This approach may provide a feasible solution, but the solution may be far from optimal one.

In order to solve Integer Linear Programming (ILP) problems, there are many proposed methods, including Branch and Bound (B&B), cutting planes, polyhedral developments, hybrid algorithms, Reformulation-Linearization Technique (RLT), facial disjunctive programming, and post-solution analysis [14]. A great deal of work has been carried out on optimization methods of the reliability problems with separable objective function and multiple constraints. Examples of these are given by Chern and Jon [4]. Most of these methods do not guarantee a global optimum solution. Lawler and Bell [11] developed an optimization method for solving Integer Nonlinear Programming (INLP) problems with zero-one variables. Their method is closely related to the 'lexicographic' method of Gilmore and Gomory [6] for the knapsack

* *Corresponding author. E-mail: sabbagh@cc.iut.ac.ir*

problem and the 'additive' algorithm of Balas [3]. Lawler and Bell assume that the objective function is a monotone non-decreasing one and the functional constraints can be expressed (rewritten) as the difference of two monotone non-decreasing functions. Sasaki *et al*. [17] developed an optimization method for solving a spare allocation problem which is applicable to integer problems with non 0-1 variables. They called their method New Lawler and Bell. Sabbagh [15,16] generalized the Lawler and Bell algorithm to solve a broader class of INLP problems. Srivastava and Fahim [19] presented a two-phase optimization procedure for integer programming problems. Aardal *et al*. [2] presented a survey of non-standard approaches to integer programming problem.

Several computational techniques (such as B&B technique, cutting planes technique, relaxation technique, outer-approximation technique etc.) which are reasonably efficient on many problems have been proposed in the literature for solving ILP problems (see, for instance, [12] and [18]). However, there is as yet no computational technique which can claim to efficiently solve many INLP problems [8].

The rest of the paper is organized as follows: Section 2 gives a description of the problem, followed by the method's motivation in Section 3. In Section 4, the authors describe their new exact procedure. They also illustrate the procedures using numerical examples. A section is reserved on detailed computational results for comparisons. The last section draws overall conclusions.

2. Description of the problem

The general nonlinear integer programming problem, with bounded integer variables, can be stated as:

Maximize $f(X) = f_1(X) - f_2(X)$

Subject to:

$$g_i(X) = g_{i1}(X) - g_{i2}(X) \le b_i, \quad i = 1,...,m,$$
$$X \in S, \tag{1}$$
$$S = \{X \mid 0 \le l_j \le x_j \le u_j, x_j \in Z^+, j = 1,...,n\}.$$

where, $X = (x_1, \cdots, x_n)$ is the design vector, $f(X)$ is the objective function, $g_i(X)$ is the i^{th} functional constraint and l_j *and* u_j are the integer lower and upper bound on the j^{th} element x_j, Z^+ is the set of non-negative integers, and b_i is the resource constant for the i^{th} constraint. It should also be noted that satisfying the design vector X upper and lower bound constraints does not imply satisfying the functional constraints $g_i(X)$.

3. Motivation for the proposed method

For very small optimization problems one can find *a global optimal integer solution* by doing a full enumeration of its integer points in the *n-dimensional cube* defined by the bounds of the variables. There are $N = \prod_{j=1}^{n}(u_j - l_j + 1)$ such integer points. The following *nested loop* generates all those *N* points.

DO $x_1 = u_1$ *to* l_1
⋱
DO $x_n = u_n$ *to* l_n
Begin
⋮
End

For example, doing full enumeration on the following problem with $N = 3\times2\times3 = 18$ points in its cube, see Column 1 of Table 1, we get $X = (1,0,2)$ with $f(X) = 11$ as a global optimal integer solution.

Maximize $f(x) = 5x_1^2 - 2x_2 + 3x_3$

Subject to:

$$2x_1 + x_2 - x_3 \le 2,$$
$$x_1 + x_2 + x_3 \le 3, \tag{2}$$
$$x_1 = 0,1,2, \quad x_2 = 0,1, \quad x_3 = 0,1,2.$$

Now let us try to solve the following problem by full enumeration.

$$\text{Max } f(x) = 5(x_1 + x_3)^3 + 2^{0.1(x_2+x_3)} - 3x_1x_2x_3$$
$$-4x_4^2 + 2^{0.1(x_5+x_6)} - 2x_7x_8$$

Subject to:

$$8x_1 - 2x_2 + 3x_3 - x_4 + 2x_5 + x_6 - x_7 - x_8 \le 500,$$

$$x_1^2 - x_2^2 + x_3^2 - x_4^2 + x_5^2 + x_7^2 - x_6 x_8 \le 1000,$$

$$(x_1 + x_2 + x_3 + x_4)(x_5 + x_6 + x_7 + x_8) \le 3000,$$

$$x_j = 0,...,50, \qquad j = 1,...,8. \tag{3}$$

This problem has *N* points in its cube to be enumerated an N is calculated as follows:

$$N = \prod_{j=1}^{8} (50+1) = 45{,}767{,}944{,}570{,}401. \tag{4}$$

Let's say that our PC, on the average, can enumerate about 1,500,000 of those points per second then it takes about 12 months or a full year to do full enumeration on this problem. However if we use implicit enumeration, as is shown later on, we obtain a global optimal solution for it in seconds. Of course the amount of time saved varies from problem to problem and depends, amongst other things, on the order of variables, see [11,15] for details. Nevertheless, for some problems like the given example the time saved in solving a problem can be drastic.

4. Description of the proposed method for solving constrained integer problems

4.1 First let us make the necessary definitions

As is done in the Lawler and Bell method, we define *the vector partial order relation* where:

$$X \le Y \Rightarrow x_j \le y_j, \qquad j = 1,...,n. \tag{5}$$

We expand the numerical order relation so that for any *X* we define *n(X)* as:

$$\begin{aligned} n(X) = {} & (x_1 - l_1)\prod_{j=2}^{n}(u_j - l_j + 1) \\ & + (x_2 - l_2)\prod_{j=3}^{n}(u_j - l_j + 1) + ... \\ & + (x_{n-1} - l_{n-1})\prod_{j=n}^{n}(u_j - l_j + 1) + (x_n - l_n). \end{aligned} \tag{6}$$

Here vector *X* is a "*general base number*", like a number in the nonmetric measurement systems, and $n(X)$ represents the numerical order of that *X*. In *general base numbers* each position of the number can have its own range.

When all components of vector *X* can take values from 0 to 9 then we can look at *X* as a decimal number. Let:

$$S = \{X = (x_1,\ldots,x_n) \mid x_j = 0,1,\ldots,u_j,\ j = 1,\ldots,n\}.$$

If for $\forall X \in S, \forall Y \in S$ in which $X \ge Y$ implies $g(X) \ge g(Y)$ then the function $g(.)$ is called *discrete isotone non-decreasing function* on *S*.

Let $V \ge 0$ be an n-vector of integers. It is easy to prove that $g(.)$ is a discrete isotone non-decreasing function on S *iff* $h(X,V) = g(X+V) - g(X) \ge 0$, for all V and *X* such that $X \in S$ and $(X+V) \in S$.

Notice that a function that is not an isotone non-decreasing function may be a discrete isotone non-decreasing function. For example, $f(X) = x_1^2 - x_1 + x_2^2$ is not an isotone non-decreasing function on $T = \{(x_1, x_2) \mid x_1 \ge 0, x_2 \ge 0\}$ because for $X = 0$, we have $f(X) = 0$ and for $X = (1/2, 0)$, we have $f(X) = -1/4$. However, this is a discrete isotone non-decreasing function on the set *S*.

Lexicographic (alphabetical or complete) ordering: Let $Y = (y_1,\ldots,y_n), Y \in R$. A vector *Y* is said to be *lexicographically positive*, written $Y >^L 0$, if $y_1 = y_2 = \ldots = y_{j-1} = 0$ and $y_j > 0$, for some $j = 1,...,n$. For $\forall X \in S, \forall Y \in S$ we write $Y >^L X$, to mean $(Y - X) >^L 0$.

We write $X \le^L Y$ to mean either (a) $(Y - X) >^L 0$, or (b) $X = Y$. We say "*X precedes Y*" (in the lexicographic ordering) to mean $Y >^L X$.

The set *S*, defined earlier, has $N = \prod_{j=1}^{n}(u_j + 1)$ members. The lexicographic ordering allows us to uniquely order the *N* members of *S* as $S_1,\ldots,S_N$ such that $S_1 = U >^L S_2 >^L S_3,\ldots,>^L S_{n-1} >^L S_N = L$ and $n(S_1) > n(S_2) > n(S_3),... > n(S_{n-1}) > n(S_N)$.

The adaptation of the Lawler and Bell algorithm requires the calculation of:

- X^-, the first vector after X in the lexicographic ordering or the numeric ordering. If $X = S_k$, $k = 1,...,N-1$ then $X^- = S_{k+1}$, and if $X = S_N = L$, we say its X^- does not exist. To get X^- subtract 1 from X.
- $\tilde{X}$, the first vector that comes after the vector X according to the lexicographic ordering but which is not in relation with X by the partial ordering. If for a particular X all the vectors after it have partial relation with X then we say that its $\tilde{X}$ does not exist.
- $\hat{X}$, the vector immediately coming before $\tilde{X}$ in the lexicographic ordering. That is to say $\hat{X} \in S$ is the last vector in the lexicographic ordering such that for every $Y \in S$ in which $X \geq^L Y \geq^L \hat{X}$ we have partial ordering i.e., $X \geq Y \geq \hat{X}$. The first vector after $\hat{X}$ in the lexicographic ordering, if it does exist, was called $\tilde{X}$. This means that for every $X \in S$ its $\hat{X}$ always exists but if $\hat{X} = L$ then there is no vector after that, that belongs to S so in this case its $\tilde{X}$ does not exist. The vector $\hat{X}$ can be used to obtain useful function bounds for the constraints of the form $g_i(X) \leq b_i$.

To calculate the above vectors, we designate by X_tilde the algorithm that calculates $\tilde{X}$, X_hat the algorithm that calculates $\hat{X}$, and X_minus the algorithms that calculates X^-. These algorithms are given below:

The generating algorithms for X_hat (see [11, 15, 16 and 17]). We first set $\hat{X} = X$. Then, starting from the rightmost vector component, $\hat{x}_n$, find the first component which is not equal to its upper bound, and then set this component and all the components to its right equal to their lower bounds. If all the components are equal to their upper bounds, i.e., if $X = U$ then $\hat{X} = L$.

The generating algorithms for X_tilde (see [11, 15, 16 and 17]). We first set $\tilde{X} = X$. Then, starting from the rightmost vector component, $\tilde{x}_n$, find the first component which is not equal to its upper bound, and, if possible, deduct 1 from its first left neighbor, and then set all the components to the right of the last changed component, equal to their upper bounds. Notice that, $\tilde{X}$ does not exist if either $X = U$ or the deduction can not be made.

The generating algorithms for X_minus (see [11, 15, 16 and 17]). Subtract 1 from X: We first set $X^- = X$. Then, if $x_n^- > l_n$ then let $x_n^- \leftarrow x_n^- - 1$ otherwise if $\tilde{x}$ does exist let $X^- = \tilde{X}$, otherwise X^- does not exist. For example, consider model (2) again with

$$S = \{X = x_1, x_2, x_3) \,|\, x_1 = 0,1,2 \;\; x_2 = 0,1, \;\; x_3 = 0,1,2\}.$$

This set has $N = 3 \times 2 \times 3 = 18$ members. For each member of S its X^-, $\hat{X}$ and $\tilde{X}$ are given in the Table 1. Notice that one can generate $\hat{X}$ and $\tilde{X}$ indirectly by doing the following operations:

$$\begin{cases} \hat{X} = \tilde{X} + 1 \Rightarrow \tilde{X} = \hat{X} - 1 \; if \; \tilde{X} \; exists \\ \hat{X} = L \; otherwise \end{cases} \tag{7}$$

4.2. The algorithm

Let us again consider the model (III) optimization problem. Here, we assume $f_k(.)$ and $g_{ik}(.)$, $i = 1,...,m, k = 1,2$ are discrete isotone non-decreasing functions. Notice that some of those functions can be zero. We can use the algorithm presented in this paper to find a global optimal integer solution for such a problem without having to check every individual point in its n-dimensional cube. In general, the more points we skip over the more efficient the algorithm becomes. As we proceed through the list of vectors we keep a record of "$\overline{X}$", the most profitable solution, and $\overline{f} = f(\overline{X})$ its most profitable value. The following rules indicate conditions under which certain vectors in the lexicographic ordering can be skipped over. The vector $X \in S$ is the one currently being examined and keep in mind that the functions $f_k(.)$ and $g_{ik}(.)$ are discrete isotone non-decreasing functions. Initially set $X = U$ and $\overline{f} = -\infty$. The rules of the algorithm are (see [11,15, 16 and 17]):

Rule 1. If $f_1(X)-f_2(\hat{X})\le \bar{f}$ then skip to $\tilde{X}$ and return to rule 1, otherwise, go to rule 2.

Justification. Clearly, for any $Y\in S$ such that:

$$\begin{aligned} & X\ge Y\ge \hat{X} \Rightarrow ((f_1(X)\ge f_1(Y)\ge f_1(\hat{X})) \\ & \&\,(-f_2(\hat{X})\ge -f_2(Y)\ge -f_2(X)) \\ & \Rightarrow \bar{f}\ge f_1(X)-f_2(\hat{X})\ge f_1(Y)-f_2(Y) \\ & \qquad \ge f_1(\hat{X})-f_2(\hat{X}). \end{aligned} \tag{8}$$

Therefore, no better solution will be found between X and $\hat{X}$ so we can safely skip to $\tilde{X}$.

Rule 2. If $\exists i \ni g_{i1}(\hat{X})-g_{i2}(X)>b_i$ then skip to $\tilde{X}$ and go to rule 1, otherwise, go to rule 3.

Justification. Clearly
$g_{i1}(Y)-g_{i2}(Y)\ge g_{i1}(Y)-g_{i2}(X)\ge g_{i1}(\hat{X})-g_{i2}(X)>b_i$ for any $Y\in S$ such that $X\ge Y\ge \hat{X}$. Therefore, no new vector between X and $\hat{X}$ will be found such that this i^{th} constraint will be satisfied, and we can safely skip to $\tilde{X}$.

Table 1. All integer points of model (2).

X	X^-	$\hat{X}$	$\tilde{X}$
(2,1,2)	(2,1,1)	(0,0,0)	Does not exist
(2,1,1)	(2,1,0)	(2,1,0)	(2,0,2)
(2,1,0)	(2,0,2)	(2,1,0)	(2,0,2)
(2,0,2)	(2,0,1)	(2,0,0)	(1,1,2)
(2,0,1)	(2,0,0)	(2,0,0)	(1,1,2)
(2,0,0)	(1,1,2)	(2,0,0)	(1,1,2)
(1,1,2)	(1,1,1)	(0,0,0)	Does not exist
(1,1,1)	(1,1,0)	(1,1,0)	(1,0,2)
(1,1,0)	(1,0,2)	(1,1,0)	(1,0,2)
(1,0,2)	(1,0,1)	(1,0,0)	(0,1,2)
(1,0,1)	(1,0,0)	(1,0,0)	(0,1,2)
(1,0,0)	(0,1,2)	(1,0,0)	(0,1,2)
(0,1,2)	(0,1,1)	(0,0,0)	Does not exist
(0,1,1)	(0,1,0)	(0,1,0)	(0,0,2)
(0,1,0)	(0,0,2)	(0,1,0)	(0,0,2)
(0,0,2)	(0,0,1)	(0,0,0)	Does not exist
(0,0,1)	(0,0,0)	(0,0,0)	Does not exist
(0,0,0)	Does not exist	(0,0,0)	Does not exist

Rule 3. If
$f(x)=f_1(X)-f_2(X)>\bar{f}\ \&\ \forall i, g_{i1}(X)-g_{i2}(X)\le b_i$,
let $\bar{X}=X$, $\bar{f}=f(X)$. Skip to X^- and go to rule 1.

Justification. Clearly if the conditions hold then the current X is a better feasible solution. Also, X^- might be a better feasible solution. Notice that whenever we say continue the enumeration with X^- or $\tilde{X}$ but there is no such vector, it means that the enumeration is complete and we should stop the calculations.

4.2.1 Numerical examples

Example 1. Let's solve model (2) that its 18 points are given in the Table 1. First we transform it into the format of model (1) as:

$$\begin{aligned} \text{Max } f(x) &= 5x_1^{\,2}-2x_2+3x_3 \\ &= 5x_1^{\,2}+3x_3-2x_2 = f_1(X)-f_2(X) \end{aligned}$$

Subject to:

$$\begin{aligned} & g_1(X)=g_{11}(X)-g_{12}(X)=2x_1+x_2-x_3\le 2, \\ & g_2(X)=x_1+x_2+x_3\le 3, \\ & x_1=0,1,2,\quad x_2=0,1,\quad x_3=0,1,2., \end{aligned} \tag{9}$$

We now apply our algorithm on this problem and get $\bar{X}(1,0,2)$ with $f(\bar{X})=11$ as a global optimal integer solution.

Example 2. Let solve model (3). Fist we transform it into the format of model (1). Let

$$\begin{aligned} & \text{Max } f(x)=(5(x_1+x_3)^3+2^{0.1(x_2+x_3)}+2^{0.1(x_5+x_6)}) \\ & -(3x_1x_2x_3+4x_4^{\,2}+2x_7x_8)=f_1(X)-f_2(X) \end{aligned}$$

Subject to:

$$\begin{aligned} & g_{11}(X)=8x_1+3x_3+2x_5+x_6, \\ & g_{12}(X)=2x_2+x_4+x_7+x_8, \\ & g_{21}(X)=x_1^{\,2}+x_3^{\,2}+x_5^{\,2}+x_7^{\,2}, \\ & g_{22}(X)=x_2^{\,2}+x_4^{\,2}+x_6x_8, \end{aligned} \tag{10}$$

$g_{31}(X) = g_3(X)$,

$g_{32}(X) = 0$.

Then, this model is in the form of model (1). If we apply the algorithm on this problem we get $\overline{X} = (28,0,29,0,0,26,0,26)$ with $f(\overline{X}) = 925978.5271$ as a global optimal integer solution in 23.84 seconds.

Let us use GAMS\BARON software (GAMS IDE 2.0.32.15) to solve this problem in two modes:

- As an NLP problem to get $X^* = (28.919, 0, 28.919, 0, 0, 25.935, 0, 25.934)$ with $f(X^*) = 967410.3127..$
- As an INLP problem to get $\tilde{X} = (28, 0, 28, 0, 0, 25, 0, 25)$ with $f(\tilde{X}) = 878092.6213$.

These results show that our integer solution is far better than the integer solution returned by GAMS\BARON commercial software.

4.3. Linear speedup: Taking advantage of the linear constraints

The algorithm of Section 4.2 starts with $X = U$ and moves forward toward $X = L$ until all the points in the cube are enumerated either directly or indirectly (implicitly). As was mentioned earlier the more points we are allowed to skip over in the cube the more efficient the algorithm becomes. The presence of linear constraints in the model can enable us to skip over more points with far fewer computations. To elaborate consider a typical linear constraint, $g(X) = \sum_{j=1}^{n} a_j x_j \le b$, and define $d = \sum_{j=1}^{n} a_j x_j - b$.

If $d > 0$ then the constraint is not satisfied at the current X and we move toward the largest vector after X, in the lexicographical ordering, that satisfies it, thus implicitly enumerating all the infeasible points in between these 2 points. The algorithm that we give is an exact algorithm not a heuristic one. It is based on the fact that: if a constraint is not satisfied at the current X, then to get the largest vector after X, either in lexicographical order or numeric order, we should change the vector X, starting from its rightmost component i.e., x_n and observe the following rules:

- If a_n is negative, then increasing x_n will decrease *d,* while its reduction will increase *d*. which is counter productive.
- If a_n is zero then changing x_n will not affect *d*. To insure we get the largest vector after *X,* in the lexicographical ordering, let x_n be equal to its upper bound.
- If a_n is positive, then increasing x_n will increase *d,* while its reduction will decrease it. When decreasing x_n, make sure its lower bound is not violated and you deduct the minimum integer amount required. If deducting the minimum integer amount required means decreasing the left side of the constraint more than *d* units, then starting from the component to the right of the current one, if possible, try to add the shortage amount.

To see the importance of it, consider the following constraint:

$$g(X) = \sum_{j=1}^{8} a_j x_j = 8x_1 - 2x_2 + 3x_3 - x_4 + 2x_5 + x_6 - x_7 - x_8 \le 23, \quad x_j = 0,...,9, \ j = 1,...8. \quad (11)$$

If we start with $X = U$ and use the algorithm of Section 4.2, after applying rules 2 and 3 totals of 144 times we get $X = (8\ 9\ 1\ 9\ 0\ 1\ 9\ 9)$ as the largest vector in lexicographical ordering that satisfies the above constraint.

Starting at $X = U$, we can get the same *X*, more efficiently, by doing the followings:

- Compute $d = g(U) - 23 = 58$ and start with x_8.
- Since a_8 is negative let keep $x_8 = 9$ and start with x_7.
- Since a_7 is negative let keep $x_7 = 9$ and start with x_6
- Since a_6 is positive, let $x_6 = 0$ and $d = 58 - 9 = 49$ and start with x_5.

- Since a_5 is positive, let $x_5 = 0$ and $d = 49 - 2\times 9 = 31$ and start with x_4.
- Since a_4 is negative, let keep $x_4 = 9$ and start with x_3.
- Since a_3 is positive, let $x_3 = 0$ and $d = 31 - 3\times 9 = 4$ and start with x_2.
- Since a_2 is negative, let keep $x_2 = 9$ and start with x_1.
- Since a_1 is positive, let $x_1 = 8$ and $d = 4 - 8\times 1 = -4$.
- Since $d = -4$, this means we have decreased the left side of the constraint 4 units more than the minimum of 58. Thus start with x_2 and, if possible, try to add 4 units. Let $d = -(-4) = 4$.
- Since a_2 is negative, start with x_3.
- Since $a_3 = 3$, let $x_3 = \lfloor 4/3 \rfloor = 1$ and $d = 4 - 3\times 1 = 1$ and start with x_4.
- Since a_4 is negative, start with x_5.
- Since $a_5 = 2$ but $\lfloor 1/2 \rfloor = 0$, start with x_6.
- Since $a_6 = 1$, let $x_6 = \lfloor d/1 \rfloor = 1$ and $d = 1 - 1\times 1 = 0$ and stop.

Thus, we started with $X = U$ and ended with $X = (8\ \ 9\ \ 1\ \ 9\ \ 0\ \ 1\ \ 9\ \ 9)$. As a result, we skipped about 10,809,800 infeasible points that are between those points.

The operations are as follows. Let's denote all the linear constraints of the model by $AX \le b$ in which A is a $k \times n$ matrix. Check the first linear constraints at the current X : If it is satisfied go to the next constraint, otherwise find the largest vector after X , in the lexicographical ordering, that satisfies it. Let that X , to be the new current X and continue until all the linear constraints are satisfied at the current X .

4.3.1. Numerical examples.

Example 1. Let's solve model (2) that was solved in 4.2.1. The model has two linear constraints. So we start with $X = U$ and we get $\overline{X} = (1,0,2)$ with $f(\overline{X}) = 11$ as a global optimal integer solution.

Example 2. Let solve model (3) that its full enumeration requires about one year execution time and using the algorithm of Section 4.2 requires 23.84 seconds and taking advantage of its only linear constraints the execution time becomes 1.57 seconds. Thus the execution time of this problem is reduced more than 15 times.

5. Computational results

Linear Speedup was coded in Visual C++ and executed on a Pentium IV computer with 2.8 GHz processor and 512 MB of RAM under Windows XP 2002. The algorithm was tested on three different sets of benchmark instances that are given in the Appendix:

Set A. The 15 benchmark hard equality constrained integer knapsack instances generated by Aardal and Lenstra [1]. Each instance contains between 5 and 10 variables. De Loera *et al*. [5] have reported that the commercial solver CPLEX 6.6 could not solve 14 of these problems in two hours.

We tried to solve some of those problems including the first instance that has 5 variables by the commercial solvers CPLEX 10 and LINGO 10 but did not succeed to get even a feasible solution in two hours. We have selected these problems because they are challenging. We refer to Table 2 in Appendix A, for the data used here. Their form is maximize $z = cx$ subject to $ax = b, x \ge 0$, $x \in Z^d$, where $b \in Z$ and where $a \in Z^d$ with gcd (a_1, . . . , a_d) = 1. For the cost vector *c*, we took the first d entries of the vector:

c=(213,-1928,-11111,-2345,9123,-12834,-123, 122331,0,0).

Set B. The 2 largest problems solved by Mohan and Nguyen [13]. The first instance has 40 variables and 3 constraints and the second instance has 100 variables and 2 constraints.

Set C. A bridge system [8].

5.1. Results on benchmark instances

Set A. We have transformed this model into the format of model (1) and reordered the variables such that $a_1/c_1 \ge a_2/c_2 \ge \ldots \ge a_n/c_n$ hold for the new order of the variables. Table 3 shows the results of

Linear Speedup on Aardal and Lenstra [1] benchmark instances for at most 100 CPU seconds on each problem.

From Table 3 in Appendix A, it is evident that Linear Speedup achieved the best-known solution in 14 of the 15 test problems, and a solution not very close to the best known one for the Prob2 instance.

Set B: Problem 1. The best solution reported in the source for the first problem, after 13.86 seconds, is a solution with $f_{\max} = 1030361$.

The result of Linear Speedup on this problem for 0.0001 second is $x_j = 99, j = 1,...,40$ with $f = 1352439$. This is far better than the best solution reported for this problem.

Set B: Problem 2. The best solution reported in the source for the second problem, after 45.23 seconds, is a solution with $f_{\max} = 303062432$.

The result of Linear Speedup on this problem for 10 seconds is a solution with $f = 304147783$. This also is a far better than the best known solution reported for this problem.

Set C: A bridge system. We solved this problem using the algorithm of Section 4.2. The final results are $x^* = (1,3,4,3,3)$ and $f^* = 0.999373$ in 0.0001 of a second.

6. Conclusion

This paper presents an optimization method that is very effective for solving some ILP and INLP problems. The results reported here illustrate the effectiveness and efficiency of the proposed method for solving some integer programming problems.

In all cases of challenging knapsack problems except one, the solution provided by the proposed procedure is in agreement with the results given in the reference paper. Where applicable, the results also agree with the solutions obtained using Sasaki's method.

For some INLP problems with linear constraints, the proposed procedure is far superior to Sasaki *et al.* method as far as speed is concerned. In terms of future research directions, for some ILP and INLP problems we may solve its corresponding continuous problem and then apply the Linear Speedup to search for a good integer solution in a neighborhood of that solution. The resulting solution then may be used as an initial feasible solution in other algorithm.

References

[1] Aardal, K. and Lenstra, A. K., 2002, *Hard Equality Constrained Integer Knapsacks*. Preliminary version in W.J. Cook and A.S. Schulz (eds.), Integer Programming and Combinatorial Optimization: 9th International IPCO Conference, Lecture Notes in Computer Science, 2337, Springer-Verlag, 350-366.

[2] Aardal, K., Weismantel, R. and Wolsey, L. A., 2002, Non-standard approaches to integer programming. *Discrete Applied Mathematics*, 123, 5-74.

[3] Balas, E., 1965, An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13, 517-545.

[4] Chern, M. and Jon, R., 1986, Reliability optimization problems with multiple constraints. *IEEE Trans. Reliability*. R-35 4, 431-436.

[5] De Loera, J. A., Haws, D., Hemmecke, R., Huggins, P. and Yoshida, R., 2005, A computational study of integer programming algorithms based on Barvinok's rational functions. *Discrete Optimization*, 2, 135-144.

[6] Gilmore, P. C. and Gomory, R. E., 1966, The theory and computation of knapsack functions. *Operations Research,* 14, 1045-1074.

[7] Gomory, R. E., 1963, *An All-Integer Programming Algorithm In Industrial Scheduling,* J. F. Muth and G. L. Thompson, Editors, Prentice-Hall, Englewood Cliffs, NJ Chapter 13.

[8] Ha, C. and Kao, W., 2006, Reliability redundancy allocation: An improved realization for non-convex nonlinear programming problems. *European Journal of Operational Research*, 171, 24-38.

[9] Kuo, W. and Zuo, M. J., 2002, *Optimal Reliability Modeling: Principles and Applications*. John Wiley & Sons, Hoboken, NJ.

[10] Land, A. H. and Doig, A. G., 1960, An automatic method of solving discrete programming problems. *Econometrica*, 28, 497-520.

[11] Lawler, E. L. and Bell, M. D., 1966, A method for solving discrete optimization problems. *Operations Research,* 4, 1098-1112.

[12] Linderoth, J. T. and Savelsbergh, M.W.P., 1999, A computational study of search strategies for mixed integer programming, *INFORMS Journal on Computing*, 11, 173-187.

[13] Mohan, C. and Nguyen, H. T., 1999, A controlled random search technique incorporating simulated annealing concept for solving integer and mixed integer global optimization problems.

Computational Optimization and Applications, 14, 103-132.

[14] Nemhauser, G. L. and Wolsey, L. A., 1999, *Integer and Combinatorial Optimization.* second ed., John Wiley & Sons, New York.

[15] Sabbagh, M. S., 1983, *A General Lexicographic Partial Enumeration Algorithm for the Solution of Integer Nonlinear Programming Problems.* D.Sc. Dissertation, The George Washington University, Washington, D.C.

[16] Sabbagh, M. S., 1996, A general lexicographic partial enumeration algorithm for the solution of integer nonlinear programming problems with discrete isotone non-decreasing objective function and constraints. *Amirkabir Journal of Science and Technology*, 8(32), 80-85.

[17] Sasaki, M., Kaburaki, S. and Yanagi, S., 1977, System availability and optimum spare units. *IEEE Trans. Reliability*. R-26, 182-188.

[18] Sherali, D. and Driscoll, J., 2003, Evolution and state-of-the-art in integer programming. *Journal of Computational and Applied Mathematics*, 124, 319-340.

[19] Srivastava, V. K. and Fahim, A., 2000, A two-phase optimization procedure for integer programming problems. *Computers & Mathematics with Applications*. 12, 1585-1595.

Appendix

Set A of test problems:

Table 2. The Benchmark instances generated by Aardal and Lenstra [1].

c	**213**	**-1928**	**-11111**	**-2345**	**9123**	**-12834**	**-123**	**122331**	**0**	**0**	**b**
Problem	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	
Cuww1	12223	12224	36674	61119	85569						89,643,482
Cuww2	12228	36679	36682	48908	61139	73365					89,716,839
Cuww3	12137	24269	36405	36407	48545	60683					58,925,135
Cuww4	13211	13212	39638	52844	66060	79268	92482				104,723,596
Cuww5	13429	26850	26855	40280	40281	53711	53714	67 1 4			45,094,584
Prob1	25067	49300	49717	62124	87608	88025	113673	119169			33,367,336
Prob2	11948	23330	30635	44197	92754	123389	136951	140745			14,215,207
Prob3	39559	61679	79625	99658	133404	137071	159757	173977			58,424,800
Prob4	48709	55893	62177	65919	86271	87692	102881	109765			60,575,666
Prob5	28637	48198	80330	91980	102221	135518	165564	176049			62,442,885
Prob6	20601	40429	40429	45415	53725	61919	64470	69340	78539	95043	22,382,775
Prob7	18902	26720	34538	34868	49201	49531	65167	66800	84069	137179	27,267,752
Prob8	17035	45529	48317	48506	86120	100178	112464	115819	125128	129688	21,733,991
Prob9	3719	20289	29067	60517	64354	65633	76969	102024	106036	119930	13,385,100
Prob10	45276	70778	86911	92634	97839	125941	134269	141033	147279	153525	106,925,262

Table 3. Solution values produced by linear speedup on Aardal and Lenstra [1] benchmark instances.

Problem	Optimal Solution	Optimal z	Our Solution	Our z
Cuww1	(7334 0 0 0 0)	1562142	(7334 0 0 0 0)	1562142
Cuww2	(3 2445 0 0 0 0)	-4713321	(3 2445 0 0 0 0)	-4713321
Cuww3	(4855 0 0 0 0 0)	1034115	(4855 0 0 0 0 0)	1034115
Cuww4	(0 0 2642 0 0 0 0)	-29355262	(0 0 2642 0 0 0 0)	-29355262
Cuww5	(1 1678 1 0 0 0 0 0)	-3246082	(1 1678 1 0 0 0 0 0)	-3246082
Prob1	(966 5 0 0 1 0 0 74)	9257735	(966 5 0 0 1 0 0 74)	9257735
Prob2	(853 2 0 4 0 0 0 27)	3471390	(852 1 0 0 0 4 0 25)	3186487
Prob3	(708 0 2 0 0 0 1 173)	21291722	(708 0 2 0 0 0 1 173)	21291722
Prob4	(1113 0 7 0 0 0 0 54)	6765166	(1113 0 7 0 0 0 0 54)	6765166
Prob5	(1540 1 2 0 0 0 0 103)	12903963	(1540 1 2 0 0 0 0 103)	12903963
Prob6	(1012 1 0 1 0 1 0 20 0 0)	2645069	(1012 1 0 1 0 1 0 20 0 0)	2645069
Prob7	(782 1 0 1 0 0 0 186 0 0)	22915859	(782 1 0 1 0 0 0 186 0 0)	22915859
Prob8	(1 385 0 1 1 0 0 35 0 0)	3546296	(1 385 0 1 1 0 0 35 0 0)	3546296
Prob9	(31 11 1 1 0 0 0 127 0 0)	15507976	(31 11 1 1 0 0 0 127 0 0)	15507976
Prob10	(0 705 0 1 1 0 0 403 0 0)	47946931	(0 705 0 1 1 0 0 403 0 0)	47946931

Set B of test problems:

Problem 1 (Problem 17 of source [13]).

Maximize $f(X) = 215x_1 + 116x_2 + 670x_3 + 924x_4 + 510x_5 + 600x_6 + 424x_7 + 942x_8 + 43x_9 + 369x_{10}$
$+ 408x_{11} + 52x_{12} + 319x_{13} + 214x_{14} + 851x_{15} + 394x_{16} + 88x_{17} + 124x_{18} + 17x_{19}$
$+ 779x_{20} + 278x_{21} + 258x_{22} + 271x_{23} + 281x_{24} + 326x_{25} + 819x_{26} + 485x_{27} + 454x_{28}$
$+ 297x_{29} + 53x_{30} + 136x_{31} + 796x_{32} + 114x_{33} + 43x_{34} + 80x_{35} + 268x_{36} + 179x_{37}$
$+ 78x_{38} + 105x_{39} + 281x_{40}$

Subject to:

$9x_1 + 11x_2 + 6x_3 + x_4 + 7x_5 + 9x_6 + 10x_7 + 3x_8 + 11x_9 + 11x_{10} + 2x_{11} + x_{12} + 16x_{13} + 18x_{14} + 2x_{15}$
$+ x_{16} + x_{17} + 2x_{18} + 3x_{19} + 4x_{20} + 7x_{21} + 6x_{22} + 2x_{23} + 2x_{24} + x_{25} + 2x_{26} + x_{27} + 8x_{28} + 10x_{29} + 2x_{30}$
$+ x_{31} + 9x_{32} + x_{33} + 9x_{34} + 2x_{35} + 4x_{36} + 10x_{37} + 8x_{38} + 6x_{39} + x_{40} \le 25000,$

$5x_1 + 3x_2 + 2x_3 + 7x_4 + 7x_5 + 3x_6 + 6x_7 + 2x_8 + 15x_9 + 8x_{10} + 16x_{11} + x_{12} + 2x_{13} + 2x_{14} + 7x_{15}$
$+ 7x_{16} + 2x_{17} + 2x_{18} + 4x_{19} + 3x_{20} + 2x_{21} + 13x_{22} + 8x_{23} + 2x_{24} + 3x_{25} + 4x_{26} + 3x_{27} + 2x_{28} + x_{29}$
$+ 10x_{30} + 6x_{31} + 3x_{32} + 4x_{33} + x_{34} + 8x_{35} + 6x_{36} + 3x_{37} + 4x_{38} + 6x_{39} + 2x_{40} \le 25000,$

$3x_1 + 4x_2 + 6x_3 + 2x_4 + 2x_5 + 3x_6 + 7x_7 + 10x_8 + 3x_9 + 7x_{10} + 2x_{11} + 16x_{12} + 3x_{13} + 3x_{14} + 9x_{15}$
$+ 8x_{16} + 9x_{17} + 7x_{18} + 6x_{19} + 16x_{20} + 12x_{21} + x_{22} + 3x_{23} + 14x_{24} + 7x_{25} + 13x_{26} + 6x_{27} + 16x_{28}$
$+ 3x_{29} + 2x_{30} + x_{31} + 2x_{32} + 8x_{33} + 3x_{34} + 2x_{35} + 7x_{36} + x_{37} + 2x_{38} + 6x_{39} + 5x_{40} \le 25000,$

$10 \le x_i \le 99, \quad i = 1,2,...,20, \qquad 20 \le x_i \le 99, \quad i = 21,22,...,40.$

Problem 2 (Problem 18 of source [14]).

$$\begin{aligned}\text{Maximize } f(X) = {} & 50x_1 + 150x_2 + 100x_3 + 92x_4 + 55x_5 + 12x_6 + 11x_7 + 10x_8 + 8x_9 + 3x_{10} + 114x_{11} \\ & + 90x_{12} + 87x_{13} + 91x_{14} + 581x_{15} + 16x_{16} + 19x_{17} + 22x_{18} + 21x_{19} + 32x_{20} + 53x_{21} \\ & + 56x_{22} + 118x_{23} + 192x_{24} + 52x_{25} + 204x_{26} + 250x_{27} + 295x_{28} + 82x_{29} + 30x_{30} \\ & + 29x_{31}^2 + 2x_{32}^2 + 9x_{33}^2 + 94x_{34} + 15x_{35}^3 + 17x_{36}^2 - 15x_{37} - 2x_{38} + x_{39} + 3x_{40}^4 + 52x_{41} \\ & + 57x_{42}^2 - x_{43}^2 + 12x_{44} + 21x_{45} + 6x_{46} + 7x_{47} - x_{48} + x_{49} + x_{50} + 119x_{51} + 82x_{52} \\ & + 75x_{53} + 18x_{54} + 16x_{55} + 12x_{56} + 6x_{57} + 7x_{58} + 3x_{59} + 6x_{60} + 12x_{61} + 13x_{62} + 18x_{63} \\ & + 7x_{64} + 3x_{65} + 19x_{66} + 22x_{67} + 3x_{68} + 12x_{69} + 9x_{70} + 18x_{71} + 19x_{72} + 12x_{73} + 8x_{74} \\ & + 5x_{75} + 2x_{76} + 16x_{77} + 17x_{78} + 11x_{79} + 12x_{80} + 9x_{81} + 12x_{82} + 11x_{83} + 14x_{84} + 16x_{85} \\ & + 3x_{86} + 9x_{87} + 10x_{88} + 3x_{89} + x_{90} + 12x_{91} + 3x_{92} + 12x_{93} - 2x_{94}^2 - x_{95} + 6x_{96} + 7x_{97} \\ & + 4x_{98} + x_{99} + 2x_{100}\end{aligned}$$

Subject to:

$$\sum_{i=1}^{100} x_i \le 7500,$$

$$\sum_{i=1}^{50} 10x_i + \sum_{i=1}^{100} x_i \le 42000,$$

$$0 \le x_i \le 99, \quad i = 1,2,...,100.$$

The result of Linear Speedup on the above problem for 10 seconds is:

x = (
99 99 99 99 99 99 0 0 0 0
99 99 99 99 99 99 99 99 99 99
99 99 99 99 99 99 99 99 99 99
99 0 99 99 99 99 0 0 0 99
99 99 0 19 99 0 0 0 0 0
99 99 99 99 99 99 99 99 0 99
99 99 99 99 0 99 99 0 99 99
99 99 99 99 99 0 99 99 99 99
99 99 99 99 99 0 99 99 0 0
99 0 99 0 0 99 99 13 0 0)

With f = 304147783

Set C: A bridge system (n = 5, m = 3)

Consider a set of five subsystems 1, 2, 3, 4 and 5 and suppose that they are connected as shown in Figure 1. Assume each block represents a stage (subsystem) that can have either parallel, options, or 2-out-of-n:G structure [8].

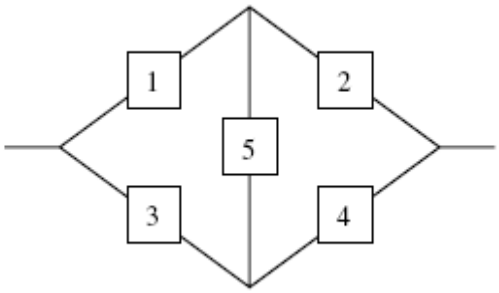

Figure 1. A bridge system.

This system is called a bridge system and it consists of five subsystems and three nonlinear and non-separable constraints. The overall system reliability $f(R_s)$ is acquired by the pivotal decomposition method [9], where $R_j = R_j(x_j)$ and $Q_j = 1 - R_j$ for all $j = 1,\ldots,5$.

Since the overall system has a complex structure, the objective function is also nonlinear and non-separable. A detailed definition of the problem appears below:

Maximize $f(R_s) = R_5(1-Q_1Q_3)(1-Q_2Q_4) + Q_5[1-(1-R_1R_2)(1-R_3R_4)]$

Subject to:

$$10\exp(\frac{x_1}{2})x_2 + 20x_3 + 3x_4^2 + 8x_5 \le 200,$$

$$10\exp(\frac{x_1}{2}) + 4\exp(x_2) + 2x_3^3 + 6[x_4^2 + \exp(\frac{x_4}{4})] + 7\exp(\frac{x_5}{4}) \le 310,$$

$$12[x_2^2 + \exp(x_2)] + 5x_3\exp(\frac{x_3}{4}) + 3x_1x_4^2 + 2x_5^3 \le 520,$$

$$(1,1,1,1,1) \le x \le (6,3,5,6,6),\ x \in Z_+^n,$$

where

$$R_1(x_1) = (0.8,0.85,0.9,0.925,0.95,0.975),$$

$$R_2(x_2) = 1-(1-0.75)^{x_2},$$

$$R_3(x_3) = \sum_{k=2}^{x_3+1}(\tbinom{x_3+1}{k})(0.88)^k(0.12)^{x_3+1-k},$$

$$R_4(x_4) = 1-(1-0.7)^{x_4},$$

$$R_5(x_5) = 1-(1-0.85)^{x_5}.$$