PAPER

# ILIN: An Implementation of the Integer Labeling Algorithm for Integer Programming

**Qiang LI**[†], **Fred JANSSEN**[††], **Zaifu YANG**[†††], *Nonmembers, and* **Tetsuo IDA**[†], *Member*

**SUMMARY** In a recent paper, Yang proposes an integer labeling algorithm for determining whether an arbitrary simplex $P$ in $R^n$ contains an integer point or not. The problem under consideration is a very difficult one in the sense that it is NP-complete. The algorithm is based on a specific integer labeling rule and a specific triangulation of $R^n$. In this paper we discuss a practical implementation of the algorithm and present a computer program (ILIN) for solving integer programming using integer labeling algorithm. We also report on the solution of a number of tested examples with up to 500 integer variables. Numerical results indicate that the algorithm is computationally simple, flexible, efficient and stable.

*key words: Simplex, integer point, integer labeling algorithm, integer linear programming*

## 1. Introduction

In the literature, much of the computational work on integer linear programming has been focusing on problems with zero-one variables. In contrast, there are relatively few numerical results on dealing with general integer programming problems. Nevertheless, Cook et al [1] have recently implemented the generalized basis reduction algorithm of Lovász and Scarf [4] and reported the solution of a number of difficult problems with up to 100 integer variables. In this paper we consider the following problem: Given an arbitrary simplex $P$, for example, the convex hull of $m+1$ $(0 \leqq m \leqq n)$ affinely independent vectors of $R^n$, determine whether $P$ contains an integer point or not. This problem is a very difficult one in the sense that it is $\mathcal{NP}$-complete shown by Yang [8]. Inspired by the work of Scarf [6] and of van der Laan and Talman [2], [3], Yang [7] develops an integer labeling algorithm to solve the problem. The algorithm is based on a specific integer labeling rule and the well-known $K_1$-triangulation of $R^n$. The main feature of the algorithm can be described as follows: The algorithm subdivides $R^n$ into $n$-dimensional simplices such that all integer points of $R^n$ are vertices of the triangulation, and then assigns an integer to each integer point of $R^n$ according to the labeling rule. Starting from an arbitrary integer point, the algorithm generates a sequence of adjacent simplices of varying dimension. Within a finite number of steps, the algorithm either finds an integer point in $P$ or (exclusively) shows that there is no integer point in $P$.

In this paper we discuss a practical implementation of this algorithm and present a computer program ILIN for solving integer programs using integer labeling algorithm. We also report a number of test examples with up to 500 integer variables. In section 2 we briefly describe the integer labeling algorithm. We discuss a practical implementation of the algorithm and present a number of test examples in section 3. Finally, we do a conclusion in section 4.

## 2. The integer labeling algorithm

The problem we consider is to test the integer feasibility of an $m$-dimensional $(0 \leqq m \leqq n)$ simplex $P$ given by

$$P = \{x \in R^n \mid a^{i\top} x \leqq b_i,\ i = 1, \cdots, n+1\},$$

where $a^{i\top} = (a_{i1}, \cdots, a_{in})$ is the $i$-th row of an $(n+1) \times n$ matrix $A$ for $i = 1, \cdots, n+1$, and $b = (b_1, \cdots, b_{n+1})^\top$ is a vector of $R^{n+1}$. Throughout the paper we make the following assumption that $a^1, \cdots, a^{n+1}$, and $b$ are integer vectors, and that the origin of $R^n$ is contained in the interior of the convex hull of the vectors $a^1, \cdots, a^{n+1}$. Notice that when $P$ is a full-dimensional simplex in $R^n$, the latter assumption is fulfilled. As usual, $Z^n$ $(Z^n_+)$ denotes the set of all (nonnegative) integer points in $R^n$. Let $N$ denote the set $\{1, \cdots, n+1\}$ and $N_{-i}$ the set $N$ without the index $i$, for $i \in N$. Now we introduce the following labeling rule.

**Labeling Rule:** Let a labeling function $l : Z^n \mapsto N \cup \{0\}$ be given as follows. To $x \in Z^n$ the label $l(x) = i$ is assigned if $i$ is the smallest index for which

$$a^{i\top} x - b_i = \max\{a^{h\top} x - b_h \mid a^{h\top} x - b_h > 0, h \in N\}.$$

If $a^{h\top} x - b_h \leqq 0$ for all $h \in N$, then the label $l(x) = 0$ is assigned to $x$.

Notice that if there exists a point $x \in Z^n$ satisfying $l(x) = 0$, then $P$ contains at least one integer point. Let $\mathcal{T}$ be the $K_1$-triangulation of $R^n$ to be described later. This simplicial subdivision of $R^n$ is such that the collection of the vertices of simplices in $\mathcal{T}$ is the set of

[†]The authors are with the Institute of Information Sciences and Electronics, University of Tsukuba, 305 Japan

[††]The author is with the Department of Econometrics, Tilburg University, 5000 LE Tilburg, The Netherlands.

[†††]The author is with the Institute of Socio-Economic Planning, University of Tsukuba, 305 Japan

all integer points of $R^n$. Throughout this paper all results are based on the $K_1$-triangulation. We denote a simplex with vertices $x^1$, $\cdots$, $x^{n+1}$ by $\sigma(x^1, \cdots, x^{n+1})$. Given an $n$-simplex $\sigma(x^1, \cdots, x^{n+1})$ in $\mathcal{T}$, let

$$L(\sigma) = \{l(x^1), \cdots, l(x^{n+1})\}.$$

An $n$-simplex $\sigma$ is called *a completely labeled (c.l.) simplex* if $|L(\sigma)| = n+1$. Specifically, an $n$-simplex $\sigma$ is called *a completely labeled simplex of type I* if $L(\sigma) = \{0\} \cup N_{-i}$ for some index $i \in N$. An $n$-simplex $\sigma$ is called *a completely labeled simplex of type II* if $L(\sigma) = N$. Observe that a completely labeled simplex of type $I$ has a vertex being an integer point in $P$.

The algorithm will be applied to any simplex $P$ for which the $(n+1) \times n$ matrix $A$ satisfies

(a) $a_{(n+1)j} \leqq 0$ for $j = 1, \cdots, n$;

(b) $a_{ii} > 0$ for $i = 1, \cdots, n$;

(c) for each $i$, $j = 1, \cdots, n$, $j \neq i$, $a_{ij} \leqq 0$ and $|a_{ij}| < a_{ii}$.

Such a formulation of the simplex $P$ is referred to as the standard form. Observe that the standard form bears many similarities with the classical Hermite normal form. A procedure for bringing any simplex into its standard form is given in Yang [7](see also Pnueli [5]). An integer point $\bar{y} \in P$ is *the greatest integer point* if $\bar{y} \geqq x$ for any $x \in P \cap Z^n$. We can easily derive the following lemma.

**Lemma 2.1:** Let a simplex $P$ be given in the standard form. If $P$ contains two integer points $x^1$ and $x^2$, it also contains the integer point

$$\bar{x} = (\max\{x_1^1, x_1^2\}, \cdots, \max\{x_n^1, x_n^2\})^\top.$$

Moreover, $P$ has a unique greatest integer point if $P$ contains at least one integer point.

Now we introduce the $K_1$-triangulation of $R^n$. Define the set $\{ q(1), \cdots, q(n+1) \}$ of $n+1$ vectors of $R^n$ by

$$q(i) = -e(i),\ i = 1, \cdots, n, \text{ and}$$
$$q(n+1) = \textstyle\sum_{i=1}^n e(i),$$

where $e(i)$ denotes the $i$-th unit vector of $R^n$, $i = 1$, $\cdots$, $n$. For a given integer $t$, $0 \leqq t \leqq n$, a $t$-dimensional simplex or a $t$-simplex, denoted by $\sigma$, is defined as the convex hull of $t+1$ affinely independent vectors $x^1$, $\cdots$, $x^{t+1}$ of $Z^n$. We usually write $\sigma = \sigma(x^1, \cdots, x^{t+1})$ and call $x^1$, $\cdots$, $x^{t+1}$ the vertices of $\sigma$. A $(t-1)$-simplex being the convex hull of $t$ vertices of $\sigma(x^1, \cdots, x^{t+1})$ is said to be a facet of $\sigma$. If $x^1 \in Z^n$ and $\pi = (\pi(1), \cdots, \pi(n))$ is a permutation of the elements of the set $\{ 1, \cdots, n \}$, then denote by $\sigma(x^1, \pi)$ the $n$-simplex with vertices $x^1$, $\cdots$, $x^{n+1}$ where $x^{i+1} = x^i + e(\pi(i))$ for each $i = 1$, $\cdots$, $n$. The $K_1$-triangulation of $R^n$ is the collection of all such simplices.

Let $v$ be an arbitrary integer point of $R^n$. The point $v$ will be the starting point of the algorithm. Define for $T$ being a proper subset of $N$ the regions $A(T)$ by

$$A(T) = \{ x \in R^n \,|\, x = v + \textstyle\sum_{j \in T} \lambda_j q(j),\ \lambda_j \geqq 0,\ j \in T \}.$$

Notice that the dimension of $A(T)$ equals $t$ with $t = |T|$. The $K_1$-triangulation subdivides any set $A(T)$ into $t$-simplices $\sigma(x^1, \pi(T))$ with vertices $x^1$, $\cdots$, $x^{t+1}$, where $x^1$ is a vertex in $A(T)$ , $\pi(T) = (\pi(1), \cdots, \pi(t))$ is a permutation of the elements of the set $T$, and $x^{i+1} = x^i + q(\pi(i))$, $i = 1$, $\cdots$, $t$. For a proper subset $T$ of $N$ a $(t-1)$-simplex $\sigma(x^1, \cdots, x^t)$, $1 \leqq t \leqq n$, is called T-complete if the $t$ vertices of $\sigma$ carry all labels of the set $T$. Note that every vertex $y$ as a zero-dimensional simplex $\{ y \}$ is $\{ l(y) \}$-complete in case $l(y) \neq 0$.

Now the algorithm generates a sequence of adjacent $t$-simplices in $A(T)$ having $T$-complete common facets. Formally the steps of the integer labeling algorithm are described as follows.

**Algorithm**

(1) Set $t = 0$, $x^1 = v$, $T = \emptyset$, $\pi(T) = \emptyset$, $\sigma =< x^1 >$, $\bar{x} = x^1$, $R_i = 0$, $i \in N$, and $Num = 1$.

(2) Calculate $l(\bar{x})$ and set $L = l(\bar{x})$. If $L = 0$, an integer point is found and the algorithm terminates. If $L$ is not an element of $T$, go to Step (4). Otherwise $L = l(x^s)$ for exactly one vertex $x^s \neq \bar{x}$ of $\sigma$.

(3) If $s = t+1$ and $R_{\pi(t)} = 0$, go to Step (5). Otherwise $\sigma$ and $R$ are adapted according to Table 1 by replacing $x^s$. Set $Num = Num + 1$. Return to Step (2) with $\bar{x}$ equal to the new vertex of $\sigma$.

(4) If $t = n$, a completely labeled simplex of type $II$ is found and the algorithm terminates. Otherwise, a $(T \cup \{ L \})$-complete simplex is found and $T$ becomes $T \cup \{ L \}$, $\pi(T)$ becomes $(\pi(1), \cdots, \pi(t), L)$, $\sigma$ becomes $\sigma(x^1, \pi(T))$, and $t$ becomes $t+1$. Set $Num = Num + 1$. Return to Step (2) with $\bar{x}$ equal to $x^{t+1}$.

(5) Let, for some $k$, $k \leqq t$, $x^k$ be the vertex of $\sigma$ with label $\pi(t)$. Then $T$ becomes $T \backslash \{ \pi(t) \}$, $\pi(T)$ becomes $(\pi(1), \cdots, \pi(t-1))$, $\sigma$ becomes $\sigma(x^1, \pi(T))$, $t$ becomes $t-1$, and return to Step (3) with $s = k$ and $Num = Num + 1$.

In the algorithm, $Num$ denotes the number of steps. In Table 1 the vector $E(i)$ denotes the $i$-th unit vector of $R^{n+1}$, $i \in N$. Without loss of generality we may assume that the algorithm is initiated at an infeasible integer point $v$. Notice that every simplex $\sigma(x^1, \pi(T))$ generated by the algorithm lies in $A(T)$ and is a $t$-simplex of the simplicial subdivision of $A(T)$ induced by the $K_1$-triangulation of $R^n$. Now in order to prove the convergence of the algorithm, we need to borrow some notions from graph theory. First, let us define a graph consisting of nodes and arcs, denoted by $G = (V, A)$. We say that a simplex $\sigma$ is a node if and only if it satisfies one of the following conditions:

| | $x^1$ becomes | $\pi(T)$ becomes | $R$ becomes |
|---|---|---|---|
| $s = 1$ | $x^1 + q(\pi(1))$ | $(\pi(2), ..., \pi(t), \pi(1))$ | $R + E(\pi(1))$ |
| $1 < s < t+1$ | $x^1$ | $(\pi(1), ..., \pi(s), \pi(s-1), \pi(s+1), ..., \pi(t))$ | $R$ |
| $s = t+1$ | $x^1 - q(\pi(t))$ | $(\pi(t), \pi(1), ..., \pi(t-1))$ | $R - E(\pi(t))$ |

**Table 1** Pivot rules if the vertex $x^s$ of $\sigma(x^1, \pi)$ is replaced.

.

(a) $\sigma = \{ v \}$;

(b) $\sigma$ is a $t$-simplex in $A(T)$ for some proper subset $T$ of $N$ with $t = |T| \geqq 1$ and at least one facet of $\sigma$ is $T$-complete.

We say that two nodes $\sigma_1$ and $\sigma_2$ in the graph $G$ are adjacent and therefore connected by an arc if and only if both $\sigma_1$ and $\sigma_2$ are in $A(T)$ for some proper subset $T$ of $N$, $t = |T|$, and one of the following cases occurs:

(a) $\sigma_1$ and $\sigma_2$ are both $t$-simplices and share a common $T$-complete facet;

(b) either $\sigma_1$ is a $T$-complete facet of $\sigma_2$ and $\sigma_2$ is a $t$-simplex or $\sigma_2$ is a $T$-complete facet of $\sigma_1$ and $\sigma_1$ is a $t$-simplex.

Observe that since the above relationship is symmetric, the arcs are not necessarily ordered. Finally, we define the degree of a node $\sigma$ in the graph by the number of nodes being connected by an arc to $\sigma$, denoted by $deg(\sigma)$. By adopting the standard argument in [2], we come to the following observation.

**Lemma 2.2:** Let $\sigma$ be a node in the graph $G$. Then

(i) $deg(\sigma) = 1$ when $\sigma = \{ v \}$;

(ii) $deg(\sigma) = 1$ when $\sigma$ is a completely labeled simplex of type $II$ or $\sigma$ has a vertex labeled with 0;

(iii) $deg(\sigma) = 2$ in all other cases.

Define a subset $C_{n+1}$ of $Z^n$ by

$$C_{n+1} = \{ x \in Z^n \mid a^{j\top} x > b_j, \text{ for } j = 1, 2, ..., n \}.$$

Now we have the following basic result obtained in [7].

**Theorem 2.3: (Test Theorem)** Let a simplex $P$ be given in the standard form. Starting with any point $v$ in $C_{n+1}$, the algorithm terminates with either the greatest integer point in $P$ or a completely labeled simplex of type $II$ indicating that there is no integer point in $P$, within a finite number of iterations.

Proof: It suffices to consider the case when $P$ contains at least one integer point. According to Lemma 2.1, $P$ has a unique greatest integer point, say $\bar{y}$. It is easy to see that $v_i > \bar{y}_i$ for all $i \in N_{-(n+1)}$. Define an $n$-dimensional cube of $R^n$ by

$$C^n = \{ x \in R^n \mid \bar{y} \leqq x \leqq 2v \}.$$

We will show that starting with $v$, the algorithm can never traverse the boundary of the set $C^n$. It is equivalent to proving that for any proper subset $T$ of $N$ there is no $T$-complete simplex $\sigma$ lying on $A(T) \cap bd(C^n)$. We need to consider the following two cases.

Case 1. For any proper subset $T$ of $N$ without element $n+1$, we will show that there is no $T$-complete simplex $\sigma$ lying on $A(T) \cap bd(C^n)$. For each $h \in T$, let

$$\begin{aligned} D(T,h) \;=\; & \{ x \in R^n \mid x_h = \bar{y}_h, \\ & \bar{y}_i \leqq x_i \leqq v_i,\ i \in T \setminus \{h\}, \\ & x_j = v_j,\ j \in N_{-(n+1)} \setminus T \}. \end{aligned}$$

Notice that $A(T) \cap bd(C^n) = \cup_{h \in T} D(T,h)$. If a simplex $\sigma$ lies on $A(T) \cap bd(C^n)$, there must exist some index $h \in T$ such that $\sigma$ is a subset of the set $D(T,h)$. For any $x \in D(T,h) \cap Z^n$, we have that $x_h = \bar{y}_h$ and for $j \neq h$, $x_j = \bar{y}_j + \beta_j$ for some $\beta_j \geqq 0$. Furthermore, since $a^{h\top}\bar{y} - b_h \leqq 0$ and $\sum_{j \neq h} a_{hj}\beta_j \leqq 0$, we have

$$\begin{aligned} a^{h\top} x - b_h &= a_{hh} x_h + \textstyle\sum_{j \neq h} a_{hj} x_j - b_h \\ &= a_{hh} \bar{y}_h + \textstyle\sum_{j \neq h} a_{hj} (\bar{y}_j + \beta_j) - b_h \\ &= a^{h\top} \bar{y} - b_h + \textstyle\sum_{j \neq h} a_{hj} \beta_j \\ &\leqq 0. \end{aligned}$$

This implies that $l(x) \neq h$. Hence there is no $T$-complete simplex on $D(T,h)$ since no vertices in $D(T,h)$ bear the label $h \in T$.

Case 2. For any proper subset $T$ of $N$ with element $n+1$, we will show that there is no $T$-complete simplex $\sigma$ lying on $A(T) \cap bd(C^n)$. For any $x \in C^n \cap Z^n$, we have that for each $j \in N_{-(n+1)}$, $x_j = \bar{y}_j + \beta_j$ for some $\beta_j \geqq 0$. Since $a^{(n+1)\top}\bar{y} - b_{(n+1)} \leqq 0$ and $\sum_{j=1}^{n} a_{(n+1)j}\beta_j \leqq 0$, we have

$$\begin{aligned} & a^{(n+1)\top} x - b_{(n+1)} && = \\ & a^{(n+1)\top} \bar{y} - b_{(n+1)} + \textstyle\sum_{j=1}^{n} a_{(n+1)j} \beta_j && \\ & && \leqq 0. \end{aligned}$$

This implies that $l(x) \neq n+1$. Hence there is no $T$-complete simplex on $A(T) \cap bd(C^n)$ since no vertices in $C^n$ carry the label $n+1 \in T$.

Now we can conclude from the above discussions that starting with $v$, the algorithm can never traverse the boundary of the set $C^n$. Since the number of simplices in $C^n$ is finite and no vertices in $C^n$ have the label $n+1$, it follows from Lemma 2.2 that within a finite number of steps, the algorithm must terminate with a simplex $\sigma^*$ with a vertex, say $w$, having the label 0. Because $\bar{y}$ is the unique vertex in $C^n$ with $l(\bar{y}) = 0$, $w$ must be equal to $\bar{y}$. This completes our proof. □

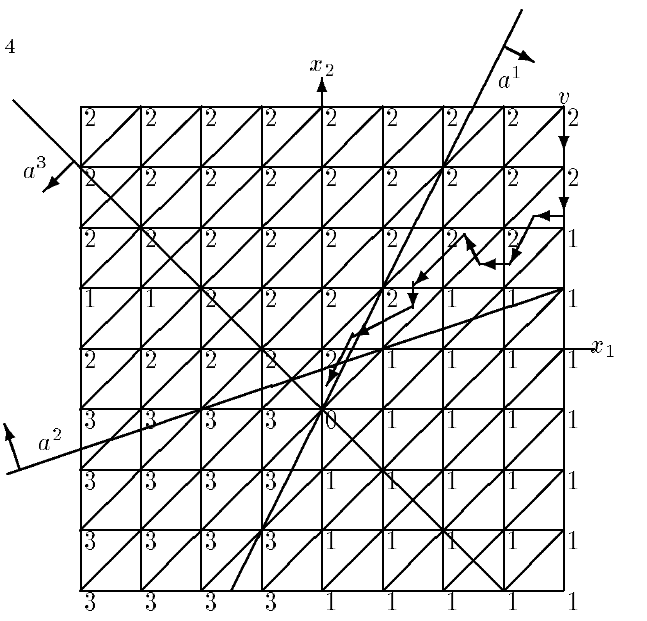

**Fig. 1** The path of the algorithm leads to an integer point in P for Example 2.5.

Let $A_{-(n+1)}$ be the matrix of $A$ without the last row and $b_{-(n+1)}$ the vector of $b$ without the last component. Let $x^R = A^{-1}_{-(n+1)} b_{-(n+1)}$. It is clear that $x^R \geqq x$ for all $x \in P$. Let $x^I = (\lfloor x_1^R \rfloor, \cdots, \lfloor x_n^R \rfloor)^\top$ where $\lfloor x \rfloor$ denotes the lower integer part of a real number $x$. It is also easy to see that $x^I \geqq y$ for all $y \in P \cap Z^n$. By sightly modifying the proof of Theorem 2.3, we have the following corollary.

**Corollary 2.4:** Let a simplex $P$ be given in the standard form. Starting with any integer point $v \geqq x^I$, the algorithm terminates with either the greatest integer point in $P$ or a completely labeled simplex of type $II$ indicating that there is no integer point in $P$, within a finite number of iterations.

From Theorem 2.3 and Corollary 2.4 we can easily see that the algorithm in fact gives an algorithm for solving the integer linear programming problem

$$\begin{array}{ll} \text{minimize} & a^{(n+1)\top} x, \\ \text{subject to} & a^{i\top} x \leqq b_i,\ i = 1, \cdots, n, \\ & x \in Z^n, \end{array}$$

under the assumption that the origin of $R^n$ is contained in the interior of the convex hull of the vectors $a^1, \cdots, a^{n+1}$.

Let us illustrate the algorithm by one example.

**Example 2.5:** The simplex is given by

$$P = \{\, x \in R^2 \,|\, a^{i\top} x \leqq b_i,\ i = 1, \cdots, 3 \,\},$$

where $a^1 = (2, -1)^\top$, $a^2 = (-1, 3)^\top$, and $a^3 = (-1, -1)^\top$, $b_1 = 1$, $b_2 = -1$, and $b_3 = 1$.
The path generated by the algorithm from $v = (4, 4)^\top$ leads to the integer point $(0, -1)^\top$ in $P$ and are shown in Figure 1.

## 3. Implementation and Experiment

ILIN (Integer Labeling algorithm for INteger programming) is an experimental software developed for solving integer programming problems by using integer labeling algorithm. It is written in C language and running on Sun UltraSparc. To be more specific, ILIN enables the user to solve the integer programming problem with which simplex P is the standard form. ILIN provides several input options: for instance, one may select some test problems existing to test the system, or select random generation to generate integer programming problems by random, or input the prepared integer programming problems that you need to solve.

In implementation of integer labeling algorithm, we first need to find a starting integral point $v \geqq x^I$ in Corollary 2.4. We use Gauss method to compute $x^R = A^{-1}_{-(n+1)} b_{-(n+1)}$, where $A_{-(n+1)}$ is the matrix of $A$ without the last row and $b_{-(n+1)}$ the vector of $b$ without the last component. Compute $x^I = (\lfloor x_1^R \rfloor, \cdots, \lfloor x_n^R \rfloor)^\top$. We can see starting with the point $x^I$, the integer labeling algorithm terminates with either the greatest integral point in simplex P or a completely labeled simplex of type $II$ indicating that there is no integral point in P, within a finite number of iterations. In practical implementation, the method starting with $x^I$ is much faster than that starting with any other integral point $v \geqq x^I$, especially for the integer programming problems with 100 or more variables.

The integer labeling algorithm is the core of our integer programming method. As we outlined in section 2, ILIN proceeds as follows(keeping in mind that we always work with the standard form). We have two functions $Label(\bar{x})$ where $\bar{x}$ is a integer point of the simplex $\sigma(x^1, \cdots, x^t)$, $1 \leqq t \leqq n$ and $Update(s, \bar{x}, \sigma)$. Function $Label(\bar{x})$ calculates $l(\bar{x})$ by using Labeling rule, if $l(\bar{x})$=0, the greatest integer point in simplex P is found and the system terminates, otherwise if t=n(n is the number of variables), a completely labeled simplex of type $II$ is found i.e. there is no solution for simplex P and the system terminates. Function $Update(s, \bar{x}, \sigma)$ updates the simplex $\sigma(x^1, \cdots, x^t)$ according to Table 1 in section 2 and assigns a new vertex of $\sigma$ to $\bar{x}$. By Corollary 2.4, it is guaranteed that the system ILIN call the two functions recursively according to the integer labeling algorithm and terminates with either the greatest integer point in simplex P or a completely labeled simplex of type $II$.

We run the system ILIN with four test problems. The first three test problems(Example 3.1-3.3) are special examples with integer matrices A and right hand sides b generated by some functions, while the fourth is random example with random integer matrices A of various sizes(ranging from 101× 100 to 501× 500) and

| $c$ | the greatest integer point or no integer solution | steps |
|---|---|---|
| $-10$ | no integer solution | 1457 |
| $-2$ | no integer solution | 1682 |
| 10 | $(0,0,0,0,0,0,0,0,0,0)$ | 1450 |
| 50 | $(0,0,0,0,0,0,0,0,0,0)$ | 1450 |
| 250 | $(31,29,26,23,21,18,15,12,9,6)$ | 2444 |
| 500 | $(136,125,113,102,90,78,65,52,39,25)$ | 2799 |
| 900 | $(294,270,246,221,195,168,141,113,84,54)$ | 2189 |
| 1000 | $(343,316,287,258,228,197,165,132,98,63)$ | 2382 |
| 2500 | $(906,833,758,680,601,519,435,348,258,166)$ | 2421 |
| 5000 | $(1869,1718,1563,1404,1240,1071,897,717,532,342)$ | 2752 |
| 7500 | $(2842,2613,2377,2135,1885,1628,1364,1091,810,520)$ | 1843 |
| 10000 | $(3811,3504,3188,2863,2528,2184,1829,1463,1086,697)$ | 2269 |

**Table 2** Tested instances of Example 3.1 by the IL algorithm for $n = 10$.

| k, c | the greatest integer point or no integer solution | steps |
|---|---|---|
| 400, 3847 | $(94,87,78,71,62,55,44,37,25,19)^T$ | 198 |
| 400, 3848 | no integer solution | 198 |
| 500, 4848 | $(118,109,99,90,78,69,56,47,32,23)$ | 184 |
| 500, 4849 | no integer solution | 184 |
| 600, 5899 | $(144,133,120,109,95,84,68,57,39,28)$ | 171 |
| 600, 5900 | no integer solution | 171 |
| 700, 6811 | $(166,153,139,126,110,97,79,65,46,32)$ | 202 |
| 700, 6812 | no integer solution | 202 |

**Table 3** Tested instances of Example 3.2 by the IL algorithm for $n = 10$.

| $n$, $k$, $c$ | steps of the IL algorithm | yes or no |
|---|---|---|
| 10, 10, 100 | 103 | no |
| 20, 100, 100 | 122 | yes |
| 30, 200, 500 | 316 | yes |
| 40, 500, 1500 | 424 | yes |
| 50, 1000, 4000 | 533 | yes |
| 60, 2000, 5000 | 388 | yes |
| 70, 4000, 9000 | 212 | yes |
| 80, 5000, 8000 | 224 | yes |
| 90, 6000, 7000 | 267 | yes |
| 100, 600, 7000 | 6228 | yes |
| 110, 400, 7000 | 8792 | yes |

**Table 4** Tested instances of Example 3.3 by the IL algorithm.

random right hand sides b. The results of the experiments will be reported as follows.

Let $P = \{x \in R^n \mid Ax \leqq b\}$ be a simplex in the standard form. Our goal is to test whether $P$ contains an integer point or not. Recall that the Fibonacci numbers are defined by $f(k+2) = f(k+1) + f(k)$, $k = 1, 2, \cdots$, with $f(1) = f(2) = 1$. We will use the Fibonacci numbers later. In the following, the integer labeling algorithm will be abbreviated to the IL algorithm. Moreover, "yes" means $P$ contains an integer point and "no" means $P$ contains no integer point. In Table 2 where $b_i = f(i) + c$, $i \in N$, we give several instances of Example 3.1 for $n = 10$. In Table 3 where $b_i = i(-1)^i n + k$, $i = 1, \cdots, n$, and $b_{n+1} = -c$, $k$ and $c$ are positive integers, we give several instances of Example 3.2 for $n = 10$.

**Example 3.1** The coefficients of $A = (a_{ij})$ are given by
$a_{ij} = -n(n+2-i) + j - 1,\ j \neq i, i, j = 1, \cdots, n,$
$a_{(n+1)j} = -j,\ j = 1, \cdots, n,$
$a_{ii} = \sum_{j \in N, j \neq i} |a_{ji}|,\ \ i = 1, \cdots, n.$

**Example 3.2** The coefficients of $A = (a_{ij})$ are given by
$a_{ij} = -(n-i+1),\ i \neq j, i, j = 1, \cdots, n,$

| $n$ | result of the IL algorithm | steps |
|---|---|---|
| 100 | the greatest integer point | 9741 |
| 200 | the greatest integer point | 26048 |
| 300 | the greatest integer point | 76303 |
| 400 | the greatest integer point | 317694 |
| 500 | the greatest integer point | 1540189 |

**Table 5** Tested instances of Example 3.4

$a_{(n+1)j} = -(n-j+1),\ j = 1, \cdots, n,$
$a_{ii} = n(n+1)/2,\ i = 1, \cdots, n.$

**Example 3.3** The coefficients of $A = (a_{ij})$ are given by
$a_{ij} = -1,\ i \neq j,\ i = 1, \cdots, n+1;\ j = 1, \cdots, n,$
$a_{ii} = n+1,\ i = 1, \cdots, n.$

In Table 4 where $b_i = i(-1)^i n + k$, $i = 1, \cdots, n$, and $b_{n+1} = -c$, $k$ and $c$ are positive integers, we give several instances of Example 3.3. We remark that this example is relatively easier than the first two examples. This is explained in Yang [7].

Example 3.4 is an instance with randomly generated integer matrices A of various sizes (ranging from 101× 100 to 501× 500) and random right hand sides b with nonnegative entries in a range between 1 and 9999. The result of the test is shown in Table 5.

**Example 3.4** The coefficients of $A = (a_{ij})$ are given by
$-10 \leqq a_{ij} \leqq -1,\ j \neq i, i, j = 1, \cdots, n,$
$-10 \leqq a_{(n+1)j} \leqq -1,\ j = 1, \cdots, n,$
$a_{ii} = \sum_{j \in N, j \neq i} |a_{ji}|,\ i = 1, \cdots, n.$

## 4. Conclusion

We have presented the system ILIN as an implementation of the integer labeling algorithm for integer programming and reported on solution of a number of tested examples with up to 500 integer variables. ILIN with its user manual and many tested examples are publicly available to the interested reader on request. We are particularly interested in having practitioners of integer programming test our program, and to provide us with feedback and suggestions. It is our belief that there is still much room for further improvements in the efficiency and applicability of the integer labeling approach.

### References

[1] W.Cook, T.Rutherford, H.Scarf and D.Shallcross, *An Implementation of the generalized basis reduction algorithm.* ORSA Journal on Computing 5(1993) 206-212.

[2] G.van der Laan and A.J.J. Talman, *A restart algorithm for computing fixed points without an extra dimension.* Mathematical Programming 17(1979)74-84.

[3] G.van der Laan and A.J.J. Talman, *A class of restart fixed point algorithms without an extra dimension.* Mathematical Programming 20(1981)33-48.

[4] L.Lovász and H.Scarf,*The generalized basis reduction algorithm.* Mathematics of Operations Research 17(1992) 751-764

[5] A.Pnueli, *A method of truncated relaxation for integer programming.* unpublished manuscript, IBM, Yorktown Heights, 1968.

[6] H.Scarf, *Production sets with indivisibilities-part I: generalities.* Econometrica 49(1981)1-32.

[7] Z.Yang, *Simplicial Fixed Point Algorithms and Applications* Ph.D. Thesis, Tilburg University, Tilburg, 1996.

[8] Z.Yang, *An integer labeling algorithm for testing the integral feasibility of arbitrary simplices*, manuscript, Tilburg University, Tilburg, 1996.

**Qiang Li** received B.S. degree and M.S. degree in computer science from Sichuan University in China in 1984 and 1987. He is now a Ph.D. student in Institute of Information Sciences and Electronics, University of Tsukuba. His research interests include parallel constraint solving and parallel optimization based on symbolic computation.

**Fred Janssen** received B.S. degree and M.S. degree in mathematics and computer science from Eindhoven University of Technology, the Netherlands. He is currently a Ph.D. student at Tilburg University, the Netherlands. His research interest includes inventory theory, queuing theory and stochastic optimization.

**Zaifu Yang** received M.S. degree in applied mathematics from Xidian University, China in 1989 and Ph.D. degree from Tilburg University, the Netherlands in 1996. He is now a post-doctor fellow of the JSPS at the University of Tsukuba. His research interests include mathematical programming, mathematical economics and mathematical finance. He has published in SIAM Journal on Control and Optimization, Mathematics of Operations Research, Journal of Mathematical Economics and some other journals.

**Tetsuo Ida** is a professor at the University of Tsukuba, where he leads a research group of symbolic computation

(SCORE) in the institute of information sciences and electronics. His research includes parallel and distributed symbolic computation, integration of functional and logic programming and term rewriting. He is an editor of the Journal of Symbolic Computation, the Journal of Functional and Logic Programming and Texts and Monographs in Symbolic Computation. He is a member of the IEICE, the IPSJ, the JSSST, the Association of Logic Programming and the IEEE Computer Society. He received a Doctor of Science from the University of Tokyo.