Dang and Ye *Fixed Point Theory and Applications* (2015) 2015:182
DOI 10.1186/s13663-015-0429-8

Fixed Point Theory and Applications
a SpringerOpen Journal

RESEARCH Open Access

A fixed point iterative approach to integer programming and its distributed computation

Chuangyin Dang[1*] and Yinyu Ye[2]

*Correspondence:
mecdang@cityu.edu.hk
[1]Department of Systems Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong, China
Full list of author information is available at the end of the article

Abstract
Integer programming is concerned with the determination of an integer or mixed-integer point in a polytope. It is an NP-hard problem and has many applications in economics and management. Although several popular methods have been developed for integer programming in the literature and extensively utilized in practices, it remains a challenging problem and appeals for more endeavors. By constructing an increasing mapping satisfying certain properties, we develop in this paper an alternative method for integer programming, which is called a fixed point iterative method. Given a polytope, the method, within a finite number of iterations, either yields an integer or mixed-integer point in the polytope or proves no such point exists. As a very appealing feature, the method can easily be implemented in a distributed way. Furthermore, the construction implies that determining the uniqueness of Tarski's fixed point is an NP-hard problem, and the method can be applied to compute all integer or mixed-integer points in a polytope and directly extended to convex nonlinear integer programming. Preliminary numerical results show that the method seems promising.

MSC: 90C10

Keywords: integer or mixed-integer point; polytope; integer programming; linear programming; self-dual embedding technique; increasing mapping; Tarski's fixed point theorem; fixed point iterative method

1 Introduction

Integer programming is concerned with the determination of an integer or mixed-integer point in a polytope. As a powerful mechanism, integer programming has been extensively applied in economics [1, 2] and management [3]. Integer programming is an NP-complete problem [4]. To solve such a problem, several methods have been developed in the literature. As an application of linear programming, the cutting plane method was pioneered in [5]. The method iteratively refines a feasible set or objective function by means of linear inequalities. The branch-and-bound method was formulated in [6]. The method gradually improves upper and lower bounds of the objective function by solving linear programs and systematically enumerates candidate solutions in branches of a tree with the full set of candidate solutions at the root by checking against the upper and lower bounds. To test whether a given feasible integer point is optimal or not, the neighborhood method was proposed in [1, 2]. The method simply checks a minimal set of points in the neighborhood of a feasible point to determine whether one of them is in the polytope or yields a better objective function value. The basis-reduction method originates in [7, 8]. As that in a branch-and-bound method, the method searches for an integer point in a polytope along a set of vectors that forms a reduced basis. The simplicial method was developed in [9, 10]. The method starts from an arbitrary integer point in the space and follows a simplicial path that either leads to an integer point in a polytope or proves no such point exists when the polytope is in a specific form. Further developments of some of these methods and new methods can be found in the recent literature such as [3, 11–18], and the references therein. These methods play an extremely important role in the development of integer programming, however, it remains a challenging problem and appeals for more endeavors. Thus, developing alternative integer programming methods is always an active research area.

Integer programming can be cast as a fixed point problem of an increasing mapping. More precisely, let $\preceq$ be a binary relation on a nonempty set S. The pair $(S, \preceq)$ is a partially ordered set if $\preceq$ is reflexive, transitive, and antisymmetric on S. A lattice is a partially ordered set $(S, \preceq)$, in which any two elements x and y have a least upper bound (supremum), $\sup_S(x, y) = \inf\{z \in S \mid x \preceq z \text{ and } y \preceq z\}$, and a greatest lower bound (infimum), $\inf_S(x, y) = \sup\{z \in S \mid z \preceq x \text{ and } z \preceq y\}$, in the set. A lattice $(S, \preceq)$ is complete if every nonempty subset of S has a supremum and an infimum in S. Let f be a mapping from S into itself. f is an increasing mapping if $f(x) \preceq f(y)$ for any x and y of S with $x \preceq y$. When $(S, \preceq)$ is a complete lattice and f is an increasing mapping, Tarski's fixed point theorem [19] asserts that f has a fixed point in S. A significant feature of Tarski's fixed point theorem is that S can be a finite set and there is no restriction on its topological structures. This feature has a profound implication for integer programming as evidenced in this paper. The computational complexity of Tarski's fixed point theorem on $(S, \leq)$ has been studied in [20], and it is polynomial-time computable if the dimension is fixed. As an application of Tarski's fixed point theorem to integer programming, an increasing-mapping approach was briefly described in [21]. However, the approach is very primitive and can only update one coordinate at each iteration.

Let $N = \{1, 2, \ldots, n\}$ and $N_0 = \{0, 1, 2, \ldots, n\}$. For x and y of R^n, $x \leq_l y$ if either $x = y$ or both $x_i = y_i$, $i = 1, 2, \ldots, k-1$, and $x_k < y_k$ for some $k \in N$, and $x \leq y$ if $x_i \leq y_i$ for all $i \in N$, where $\leq_l$ is the lexicographic order on R^n and $\leq$ is the componentwise order on R^n. Let S be any given finite set of R^n. Then $(S, \leq_l)$ is a complete lattice. We now convert integer programming into the computation of fixed points of an increasing mapping from a finite lattice into itself, which leads to the fixed point iterative method proposed in this paper and is the driving force behind our research endeavors.

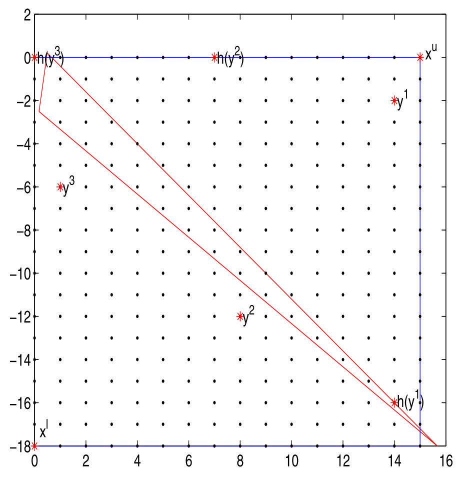
Consider $P = \{x \in R^2 \mid Ax \leq b\}$ with

$$A = \begin{pmatrix} -17 & 2 \\ 6 & 5 \\ -3 & -3 \end{pmatrix}$$

and $b = (-8, 4, 7)^\top$. This polytope is illustrated in Figure 1.

Let $D(P)$ denote the set of all the integer points with $x^l \leq x \leq x^u$ in Figure 1. The idea is to define an increasing mapping h from $D(P)$ into itself such that at least $h(x) \leq_l x$

Figure 1 An illustration of the idea.



for any $x \in D(P)$ with $x \notin P$ and $x \neq x^l$ and that $h(x^*) = x^*$ if and only if either $x^* \in P$ or $x^* = x^l$. Such a mapping is illustrated in Figure 1. This simple example stimulates the idea in this paper though the situation is far more complicated when the dimension is higher.

In this paper, with this constructing we develop a fixed point iterative method for integer programming. A self-dual technique is applied for a solution to a bounding linear program in the development. Given any polytope, within a finite number of iterations, the method either yields an integer or mixed-integer point in the polytope or proves no such point exists. Theoretically, one can make the method be a polynomial-time algorithm when the dimension is fixed. But a more appealing feature of the method is that it can easily be implemented in a distributed way. Furthermore, the construction implies that determining the uniqueness of Tarski's fixed point is an NP-hard problem, and the method can be applied to compute all integer or mixed-integer points in a polytope and directly extended to convex nonlinear integer programming. Preliminary numerical results show that the method is promising, and may offer a comparable solution to integer programming though a comprehensive comparison with the existing methods is beyond the scope of this paper.

The rest of the paper is organized as follows. A fixed point iterative method is first developed for integer programming in Section 2. Then a distributed implementation of the method and the computation of all integer points in a polytope are discussed in Section 3. Preliminary numerical results are presented in Section 4.

2 A fixed point iterative method

Let

$$P = \{x \in R^n \mid Ax + Gw \leq b \text{ for some } w \in R^p\},$$

where $A \in R^{m\times n}$ is an $m \times n$ integer matrix with $n \geq 2$, $G \in R^{m\times p}$ an $m \times p$ matrix, and b a vector of R^m. We assume throughout this paper that P is bounded and full dimensional. For a real number α, let $\lfloor \alpha \rfloor$ denote the greatest integer less than or equal to α. For $x = (x_1, x_2, \dots, x_n)^\top \in R^n$, let $\lfloor x \rfloor = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \dots, \lfloor x_n \rfloor)^\top$.

Let $x^{\max} = (x_1^{\max}, x_2^{\max}, \dots, x_n^{\max})^\top$ with $x_j^{\max} = \max_{x \in P} x_j$, $j = 1, 2, \dots, n$, and $x^{\min} = (x_1^{\min}, x_2^{\min}, \dots, x_n^{\min})^\top$ with $x_j^{\min} = \min_{x \in P} x_j$, $j = 1, 2, \dots, n$. Then $x^{\min} \leq x \leq x^{\max}$ for all $x \in P$. Let

$$D(P) = \{x \in Z^n \mid x^l \leq x \leq x^u\},$$

where $x^u = \lfloor x^{\max} \rfloor$ and $x^l = \lfloor x^{\min} \rfloor$. Thus, $D(P)$ contains all integer points in P. We assume without loss of generality that $x^l < x^{\min}$ (let $x_i^l = x_i^{\min} - 1$ if $x_i^l = x_i^{\min}$ for some $i \in N$) and that

$$x_1^u - x_1^l \leq x_2^u - x_2^l \leq \cdots \leq x_n^u - x_n^l,$$

which can be obtained by interchanging the columns of A if necessary.

For $z \in R^n$ and $k \in N_0$, let

$$P(z, k) = \{x \in P \mid x_i = z_i, 1 \leq i \leq k \text{ and } x_i \leq z_i, k + 1 \leq i \leq n\}.$$

Given an integer point $y \in D(P)$ with $y_1 > x_1^l$, we present in the following a fixed point iterative method to determine whether there is an integer point $x^* \in P$ with $x^* \leq_l y$.

Initialization: Let $y^0 = y$, $k = n - 1$, and $q = 0$.

Step 1: If $y^q \in P$ or $y^q = x^l$, *Stop*; else, go to *Step* 2.

Step 2: If $y_i^q \leq x_i^l$ for some $i \in N$ or $P(y^q, k) = \emptyset$, go to *Step* 5; else, go to *Step* 3.

Step 3: Solve the linear program

$$\max \sum_{j=k+1}^{n} x_j^j$$
$$\text{subject to } x^j \in P(y^q, k), \quad j = k + 1, k + 2, \dots, n,$$

to obtain the optimal value of x_j^j, denoted by $x_j^j(y^q)$, $j = k + 1, k + 2, \dots, n$, and go to *Step* 4.

Step 4: If $y_j^q > x_j^j(y^q)$ for some $j \geq k + 1$, let $y^{q+1} = (y_1^{q+1}, y_2^{q+1}, \dots, y_n^{q+1})^\top$ with

$$y_i^{q+1} = \begin{cases} y_i^q & \text{if } 1 \leq i \leq k, \\ \lfloor x_i^i(y^q) \rfloor & \text{if } k + 1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \dots, n$, and $q = q + 1$, and go to *Step* 1; else, let $k = k + 1$ and go to *Step* 3.

Step 5: If $k = 0$, let $y^{q+1} = x^l$ and $q = q + 1$; else, let $y^{q+1} = (y_1^{q+1}, y_2^{q+1}, \dots, y_n^{q+1})^\top$ with

$$y_i^{q+1} = \begin{cases} y_i^q & \text{if } 1 \leq i \leq k - 1, \\ y_i^q - 1 & \text{if } i = k, \\ x_i^u & \text{if } k + 1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \dots, n$, $q = q + 1$, and $k = k - 1$. Go to *Step* 1.

At each iteration, the method needs to solve a bounding linear program, which may have no feasible solution. To effectively address this issue, one can apply the self-dual embedding technique in [22, 23] or any best available software packages. The following two examples illustrate how the method works.

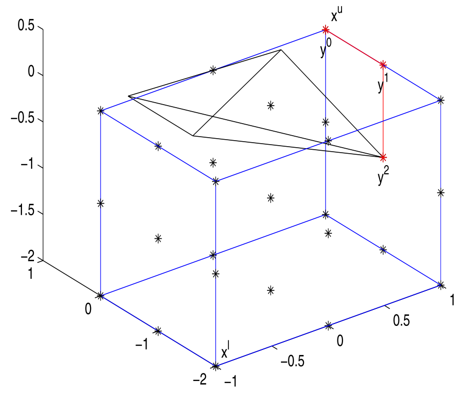


Figure 2 An illustration of the method.

Example 1 Consider $P = \{x \in R^3 \mid Ax \le b\}$ with

$$A = \begin{pmatrix} -1 & 0 & 2 \\ 0 & -2 & 1 \\ -1 & 0 & -2 \\ 1 & 1 & 0 \end{pmatrix}$$

and $b = (0,1,1,0,)^\top$. We have $x^u = (1,0,0)^\top$ and $x^l = (-1,-2,-2)^\top$.
Let $y = x^u$, $y^0 = y$, and $k = 3 - 1 = 2$.

Iteration 1: Since $P(y^0,2) = \emptyset$, we obtain from *Step* 5

$$y^1 = \left(y_1^0, y_2^0 - 1, x_3^u\right)^\top = (1,-1,0)^\top$$

and $k = k - 1 = 2 - 1 = 1$.

Iteration 2: Solving

$$\max x_2^2 + x_3^3$$
$$\text{subject to } x^j \in P\left(y^1,1\right), \quad j = 2,3,$$

we obtain $x_2^2(y^1) = -1$ and $x_3^3(y^1) = -1$. Since $y_3^1 = 0 > -1 = x_3^3(y^1)$, we obtain from *Step* 4

$$y^2 = \left(y_1^1, \lfloor x_2^2\left(y^1\right)\rfloor, \lfloor x_3^3\left(y^1\right)\rfloor\right)^\top = (1,-1,-1)^\top,$$

which is an integer point in P.

An illustration of y^0, y^1, and y^2 can be found in Figure 2.

Example 2 Consider $P = \{x \in R^3 \mid Ax \le b\}$ with

$$A = \begin{pmatrix} -4 & 3 & 2 \\ -1 & 4 & -2 \\ -1 & -5 & -1 \\ 2 & 1 & 1 \end{pmatrix}$$

and $b = (-2,0,1,1)^\top$. We have $x^u = (1,0,0)^\top$ and $x^l = (0,-1,-2)^\top$.

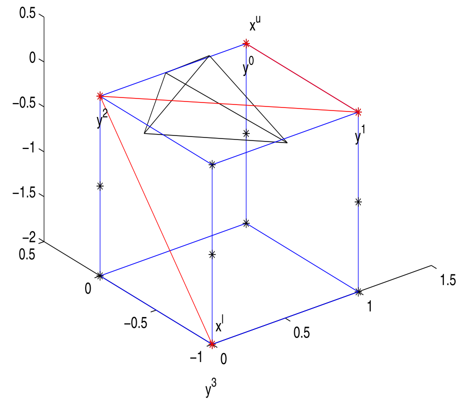


Figure 3 An illustration of the method.

Let $y = x^u$, $y^0 = y$, and $k = n - 1 = 2$.

Iteration 1: Since $P(y^0, 2) = \emptyset$, we obtain from *Step* 5

$$y^1 = \left(y_1^0, y_2^0 - 1, x_3^u\right)^\top = (1, -1, 0)^\top$$

and $k = k - 1 = 2 - 1 = 1$.

Iteration 2: Since $P(y^1, 1) = \emptyset$, we obtain from *Step* 5

$$y^2 = \left(y_1^1 - 1, x_2^u, x_3^u\right)^\top = (0, 0, 0)^\top$$

and $k = k - 1 = 1 - 1 = 0$.

Iteration 3: Since $P(y^2, 0) = \emptyset$, we obtain from *Step* 5

$$y^3 = x^l = (0, -1, -2)^\top,$$

which shows that there is no integer point in P.

An illustration of y^0, y^1, y^2, and y^3 can be found in Figure 3.

For $q = 0, 1, \ldots$, let k_q denote the value of k at which the method determines y^q. Clearly, $x^l - e \le y^q \le x^u$ with $e = (1, 1, \ldots, 1)^\top \in R^n$.

Lemma 1 *For* $q = 0, 1, \ldots$,

$$y^{q+1} \le y^q \quad or \quad y^{q+1} \le_l y^q$$

with $y^q \ne y^{q+1}$.

Proof This lemma is proved in two cases.

Case 1: *Suppose that* $P(y^q, k_q) = \emptyset$. Then the method will perform *Step* 5. If $k_q = 0$, we obtain from *Step* 5 that $y^{q+1} = x^l$, and consequently, $y^{q+1} \le y^q$ with $y^q \ne y^{q+1}$. Assume that $k_q > 0$. Then we obtain from *Step* 5 that $y^{q+1} = (y_1^{q+1}, y_2^{q+1}, \ldots, y_n^{q+1})^\top$ with

$$y_i^{q+1} = \begin{cases} y_i^q & \text{if } 1 \le i \le k_q - 1, \\ y_i^q - 1 & \text{if } i = k_q, \\ x_i^u & \text{if } k_q + 1 \le i \le n, \end{cases}$$

$i = 1, 2, \dots, n$. Thus, $y^{q+1}_{k_q} < y^q_{k_q}$. Therefore, $y^{q+1} \leq_l y^q$ with $y^q \neq y^{q+1}$.

Case 2: *Suppose that* $P(y^q, k_q) \neq \emptyset$. Then the method will repeatedly perform *Steps* 3 *and* 4 right before it goes to *Step* 1. Since $y^q \notin P$, as k increases one by one, it will reach a value k_{q+1} such that $y^q_j > x^j_j(y^q)$ for some $j \geq k_{q+1} + 1$. When this occurs, we obtain from *Step* 4 that $y^{q+1} = (y^{q+1}_1, y^{q+1}_2, \dots, y^{q+1}_n)^\top$ with

$$y^{q+1}_i = \begin{cases} y^q_i & \text{if } 1 \leq i \leq k_{q+1}, \\ \lfloor x^i_i(y^q) \rfloor & \text{if } k_{q+1} + 1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \dots, n$. Moreover, one can see from *Step* 3 that

$$x^j_j(y^q) \leq y^q_j, \quad j = k_{q+1} + 1, k_{q+1} + 2, \dots, n.$$

Therefore, $y^{q+1} \leq y^q$ with $y^q \neq y^{q+1}$. The proof is completed. □

Theorem 1 *Given an integer point* $y \in D(P)$ *with* $y_1 > x^l_1$, *the method, within a finite number of iterations, either yields an integer point* $x^* \in P$ *with* $x^* \leq_l y$ *or proves no such point exists.*

Proof Let y be any given integer point in $D(P)$ with $y_1 > x^l_1$. Suppose that $y \notin P$ and there is some integer point $z^0 \in P$ with $z^0 \leq_l y$. We assume without loss of generality that z^0 is the largest integer point of P satisfying that $z^0 \leq_l y$. Applying mathematical induction, we show in the following that $z^0 \leq_l y^q$, $q = 1, 2, \dots$.

1. Consider the case of $q = 1$. From the method, we know that $k_0 = n - 1$.

(a) *Suppose that* $P(y^0, k_0) = \emptyset$. Then the method will perform *Step* 5, and we obtain from *Step* 5 that $y^1 = (y^1_1, y^1_2, \dots, y^1_n)^\top$ with

$$y^1_i = \begin{cases} y^0_i & \text{if } 1 \leq i \leq n - 2, \\ y^0_i - 1 & \text{if } i = n - 1, \\ x^u_i & \text{if } i = n, \end{cases}$$

$i = 1, 2, \dots, n$, and $k_1 = n - 2$. If $y^0_i > z^0_i$ for some $i \leq n - 2$, then $z^0 \leq_l y^1$ with $z^0_j < y^1_j$ for some $j \leq k_1$. Assume that $y^0_i = z^0_i$ for all $i \leq n - 2$. Then, from $P(y^0, k_0) = \emptyset$ and $z^0 \leq_l y^0$, we derive that $z^0_{n-1} < y^0_{n-1}$ since otherwise $P(y^0, k_0) \neq \emptyset$. Therefore, $z^0 \leq y^1$.

(b) *Suppose that* $P(y^0, k_0) \neq \emptyset$. Then the method will perform *Steps* 3 *and* 4. Since $y^0 \notin P$, $y^0_n > x^n_n(y^0)$. Thus, we obtain from *Step* 4 $y^1 = (y^1_1, y^1_2, \dots, y^1_n)^\top$ with

$$y^1_i = \begin{cases} y^0_i & \text{if } 1 \leq i \leq n - 1, \\ \lfloor x^n_n(y^0) \rfloor & \text{if } i = n, \end{cases}$$

$i = 1, 2, \dots, n$, and $k_1 = n - 1$. If $y^0_i > z^0_i$ for some $i \leq n - 1$, then $z^0 \leq_l y^1$ with $z^0_j < y^1_j$ for some $j \leq k_1$. Assume that $y^0_i = z^0_i$ for all $i \leq n - 1$. Then we derive from $z^0 \leq_l y^0$ that $z^0 \in P(y^0, k_0)$. Thus, $z^0_n \leq \lfloor x^n_n(y^0) \rfloor$. Therefore, $z^0 \leq y^1$.

2. *Induction hypothesis*: For any given $1 \leq h \leq q$, we assume that $y^h \notin P$ and that $z^0 \leq y^h$ or $z^0 \leq_l y^h$ with $z^0_j < y^h_j$ for some $j \leq k_h$.

3. With this induction hypothesis, we prove in the following that $z^0 \leq y^{q+1}$ or $z^0 \leq_l y^{q+1}$ with $z^0_j < y^{q+1}_j$ for some $j \leq k_{q+1}$ under two cases.

Case 1: *Suppose that* $P(y^q, k_q) = \emptyset$. Then the method will perform *Step* 5. Assume that $k_q = 0$. From the method, we know that *Step* 5 must be performed at least once before y^q is generated. Let y^{q_0} be the point obtained by the method in the last performance of *Step* 5 before y^q is generated. Thus, $k_{q_0} = k_q = 0$, $k_{q_0-1} = 1$, and $P(y^{q_0-1}, k_{q_0-1}) = \emptyset$. From *Step* 5, we know that $y^{q_0} = (y_1^{q_0}, y_2^{q_0}, \ldots, y_n^{q_0})^\top$ with

$$y_i^{q_0} = \begin{cases} y_i^{q_0-1} - 1 & \text{if } i = 1, \\ x_i^u & \text{if } 2 \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. This together with $P(y^{q_0-1}, k_{q_0-1}) = \emptyset$ and the induction hypothesis for $h = q_0 - 1$ leads to that $z^0 \le y^{q_0}$. If $q = q_0$, then $z^0 \le y^q = y^{q_0}$. Suppose that $q > q_0$. Then $q = q_0 + 1$ and the method will perform *Steps* 3 *and* 4 to generate y^q right after y^{q_0} is generated. Since $k_q = k_{q_0}$, the method will perform once *Steps* 3 *and* 4 to generate y^q. With $k_q = 0$, we obtain from *Step* 4 that $y^q = (y_1^q, y_2^q, \ldots, y_n^q)^\top$ with

$$y_i^q = \lfloor x_i^i(y^{q_0}) \rfloor, \quad i = 1, 2, \ldots, n.$$

Thus, it follows from $z^0 \in P(y^{q_0}, k_{q_0})$ that $z^0 \le y^q$. Therefore, $z^0 \in P(y^q, k_q)$. It contradicts with $P(y^q, k_q) = \emptyset$. So, we must have $k_q > 0$.

From *Step* 5, we obtain $y^{q+1} = (y_1^{q+1}, y_2^{q+1}, \ldots, y_n^{q+1})^\top$ with

$$y_i^{q+1} = \begin{cases} y_i^q & \text{if } 1 \le i \le k_q - 1, \\ y_i^q - 1 & \text{if } i = k_q, \\ x_i^u & \text{if } k_q + 1 \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$, and $k_{q+1} = k_q - 1$. If $z_i^0 < y_i^q$ for some $i \le k_q - 1$, then $z^0 \le_l y^{q+1}$ with $z_j^0 < y_j^{q+1}$ for some $j \le k_{q+1}$. Suppose that $y_i^q = z_i^0$, $i = 1, 2, \ldots, k_q - 1$. Hence, $z_{k_q}^0 < y_{k_q}^q$ from the induction hypothesis since otherwise $P(y^q, k_q) \neq \emptyset$. Therefore, $z^0 \le y^{q+1}$.

Case 2: *Suppose that* $P(y^q, k_q) \neq \emptyset$. Then the method will repeatedly perform *Steps* 3 *and* 4 right before it goes to *Step* 1. Since $y^q \notin P$, as k increases one by one, it will reach a value $k_{q+1} \ge k_q$ such that $y_j^q > x_j^j(y^q)$ for some $j \ge k_{q+1} + 1$. When this occurs, we obtain from *Step* 4 that $y^{q+1} = (y_1^{q+1}, y_2^{q+1}, \ldots, y_n^{q+1})^\top$ with

$$y_i^{q+1} = \begin{cases} y_i^q & \text{if } 1 \le i \le k_{q+1}, \\ \lfloor x_i^i(y^q) \rfloor & \text{if } k_{q+1} + 1 \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. If $z_i^0 < y_i^q$ for some $i \le k_q$, then $z^0 \le_l y^{q+1}$ with $z_j^0 < y_j^{q+1}$ for some $j \le k_{q+1}$. Suppose that $y_i^q = z_i^0$ for all $i \le k_q$. Thus, $z^0 \le y^q$ from the induction hypothesis for $h = q$.

- Consider $k_{q+1} = k_q$. Since $z^0 \in P(y^q, k_q)$, $z_i^0 \le \lfloor x_i^i(y^q) \rfloor$ for all $k_{q+1} + 1 \le i \le n$. Therefore, $z^0 \le y^{q+1}$.
- Consider $k_{q+1} > k_q$. If $z_i^0 < y_i^q$ for some $k_q < i \le k_{q+1}$, then it follows from *Steps* 3 *and* 4 that $z^0 \le_l y^{q+1}$ with $z_j^0 < y_j^{q+1}$ for some $j \le k_{q+1}$. Suppose that $y_i^q = z_i^0$ for all $k_q < i \le k_{q+1}$. Since $z^0 \le y^q$, $z^0 \in P(y^q, k_{q+1})$ and consequently, $z_i^0 \le \lfloor x_i^i(y^q) \rfloor$ for all $k_{q+1} + 1 \le i \le n$. Therefore, $z^0 \le y^{q+1}$.

The above results together with mathematical induction show that

$$z^0 \le_l y^q, \quad q = 1, 2, \ldots.$$

From Lemma 1, we know that $y^{q+1} \leq y^q$ or $y^{q+1} \leq_l y^q$ with $y^{q+1} \neq y^q$. Therefore, within a finite number of iterations, the method meets z^0 since there are only a finite number of integer points in the set

$$\{z \in Z^n \mid z^0 \leq_l z \leq_l y \text{ and } x^l - e \leq z \leq x^u\}.$$

This completes the proof. □

As a corollary of Theorem 1, we come to the following conclusion.

Corollary 1 *Starting from $y^0 = x^u$, the method, within a finite number of iterations, either yields an integer point in P or proves no such point exists.*

3 Distributed computation and computing all integer points in a polytope

For any given positive integer ν, let x^i, $i = 1, 2, \ldots, \nu$, be a sequence of different integer points in $D(P)$ with $x^l \leq x^1 \leq_l x^2 \leq_l \cdots \leq_l x^\nu = x^u$. Then the method can easily be implemented in a distributed way by starting from x^i, $i = 1, 2, \ldots, \nu$, simultaneously.

The method can also be applied to compute all integer points in P, which is as follows.

Step 0: Use the method starting from x^u to compute an integer point in P. If no integer point has been found, *Stop*. Otherwise, let s^1 be the solution found by the method and $g = 1$, and go to *Step* 1.

Step 1: Let $y^0 = (y_1^0, y_2^0, \ldots, y_n^0)^\top$ with

$$y_i^0 = \begin{cases} s_i^g & \text{if } i < n, \\ s_i^g - 1 & \text{if } i = n, \end{cases}$$

$i = 1, 2, \ldots, n$, and go to *Step* 2.

Step 2: If $y^0 \in P$, let $s^{g+1} = y^0$ and $g = g + 1$, and go to *Step* 1. Otherwise, go to *Step* 3.

Step 3: Use the method starting from y^0 to compute an integer point in P. If no integer point has been found, *Stop*. Otherwise, let s^{g+1} be the solution found by the method and $g = g + 1$, and go to *Step* 1.

4 Numerical results

In this section, we apply the method to determine whether there is an integer point in the polytope of the market split problem and the polytope of the 0-1 knapsack feasibility problem though a comprehensive comparison with the existing methods is beyond the scope of this paper. The method has been coded in C++ and run on a workstation of Lenovo ThinkStation D20 4155-BM4 with 16 processors. In our implementation of the method, each linear program is solved by the linear program solver of ILOG CPLEX with all the parameter values automatically set by ILOG CPLEX itself. We have also run ILOG CPLEX on the same problem instance and found that the branch-and-cut strategy is the best of ILOG CLEX. In the presentation of numerical results, NumLPs stands for the total number of linear programs solved by the method and the branch-and-cut strategy of ILOG CLEX for each instance. In the feasibility category, 'Feasible' appears if an instance has a feasible integer point and 'Infeasible' otherwise. In our numerical experiments, to convert a problem into an equivalent problem of determining whether there is an integer point in

Table 1 The market split problem

Prob.	p	q	The method		The best CPLEX strategy	
			NumLPs	Feasibility	NumLPs	Feasibility
1	5	40	9,640	Feasible	87,136	Feasible
2	5	40	32,015	Feasible	718,883	Feasible
3	5	40	22,221	Feasible	484,796	Feasible
4	5	40	12,670	Feasible	139,967	Feasible
5	5	40	49,709	Feasible	454,552	Feasible
6	5	40	54,525	Infeasible	677,463	Infeasible
7	5	40	105,670	Infeasible	1,644,945	Infeasible
8	5	40	90,204	Infeasible	1,593,382	Infeasible
9	5	40	93,751	Infeasible	1,061,334	Infeasible
10	5	40	67,565	Infeasible	1,039,454	Infeasible
11	5	40	90,218	Infeasible	1,134,168	Infeasible
12	5	40	36,204	Feasible	857,912	Feasible
13	5	40	106,082	Infeasible	2,189,829	Infeasible
14	5	40	33,699	Infeasible	649,045	Infeasible
15	5	40	64,368	Infeasible	808,468	Infeasible
16	5	40	38,577	Feasible	370,900	Feasible
17	5	40	26,167	Feasible	162,529	Feasible
18	5	40	75,633	Feasible	96,595	Feasible
19	5	40	86,061	Infeasible	964,038	Infeasible
20	5	40	36,737	Infeasible	801,745	Infeasible
21	5	40	67,556	Infeasible	881,563	Infeasible
22	5	40	16,170	Feasible	1,136,200	Feasible
23	5	40	33,848	Feasible	225,732	Feasible
24	5	40	78,172	Infeasible	784,658	Infeasible
25	5	40	75,375	Infeasible	1,412,998	Infeasible

a full-dimensional polytope given by $P = \{x \in R^n \mid Ax \leq b\}$, we apply the basis-reduction algorithm of [24] in the same way as that in [11] and in the appendix with $N_1 = 10{,}000$ and $N_2 = 100{,}000$.

Example 3 (The market split problem) The market split problem given in [15] is to determine whether the system, $Cx = d$, has a 0-1 integer solution, where $C = (c_{ij})$ is a $p \times q$ (*e.g.*, $q = 10(p-1)$) nonnegative integer matrix and $d = (d_1, d_2, \ldots, d_p)^\top$ is an integer vector given by $d_i = \lfloor \sum_{j=1}^{n} c_{ij}/2 \rfloor$, $i = 1, 2, \ldots, p$. In our numerical experiments, $c_{ij} \in [0, 99]$, $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, q$, are generated randomly.

For the problem with $p = 5$ and $q = 40$, we have solved 25 instances using the method and the best CPLEX strategy. Numerical results for 25 instances of the problem are given in Table 1.

To demonstrate the capability of distributed computation of the method, we have implemented the method in a distributed way to solve the market split problem with $p = 6$ and $q = 50$. We divide the problem space into 32 parts and run 16 subproblems simultaneously on the workstation. In the presentation of numerical results, MAX NumLPs stands for either the largest number of linear programs consumed by the method for any of the 32 subproblems when an instance is infeasible or the smallest number of linear programs consumed by the method for the subproblem in which a feasible solution is found. Numerical results for five instances of the problem are given in Table 2.

Example 4 (The 0-1 knapsack feasibility problem) Find a 0-1 solution of $p^\top x = d$, where $p = (p_1, p_2, \ldots, p_{n+1})^\top > 0$ and $p_i \neq p_j$ for all $i \neq j$. In our numerical experiments, $p_j \in [10^2, 10^4]$, $j = 1, 2, \ldots, n+1$, and $d \in [10^2, 10^4]$ are generated randomly. Numerical results of the method for this problem are given in Table 3.

Table 2 The market split problem

Prob.	p	q	NumLPs	Feasibility
1	6	50	1.25E+07	Feasible
2	6	50	4.13E+07	Infeasible
3	6	50	3.46E+07	Feasible
4	6	50	4.17E+07	Feasible
5	6	50	3.73E+07	Feasible

Table 3 The 0-1 knapsack feasibility problem

Prob.	n	The method		The best CPLEX strategy	
		NumLPs	Feasibility	NumLPs	Feasibility
1	1,000	1,002	Feasible	3,023	Feasible
2	1,000	1,010	Feasible	1,542	Feasible
3	1,000	1,027	Feasible	1,495	Feasible
4	1,000	1,011	Feasible	1,428	Feasible
5	1,000	1,118	Feasible	883	Feasible
6	1,000	1,035	Feasible	2,023	Feasible
7	1,000	1,000	Infeasible	1,280	Infeasible
8	1,000	1,002	Feasible	1,360	Feasible
9	1,000	1,002	Feasible	998	Feasible
10	1,000	1,013	Feasible	1,087	Feasible
11	1,000	1,321	Feasible	1,577	Feasible
12	1,000	1,003	Feasible	1,117	Feasible
13	1,000	1,024	Feasible	1,638	Feasible
14	1,000	1,005	Feasible	1,122	Feasible
15	1,000	1,019	Feasible	1,097	Feasible
16	1,000	1,007	Feasible	1,365	Feasible
17	1,000	999	Infeasible	1,315	Infeasible
18	1,000	1,572	Feasible	3,741	Feasible
19	1,000	1,031	Feasible	1,170	Feasible
20	1,000	1,015	Feasible	2,702	Feasible
21	1,000	1,002	Feasible	978	Feasible
22	1,000	1,007	Feasible	1,486	Feasible
23	1,000	1,005	Feasible	1,043	Feasible
24	1,000	1,001	Feasible	3,643	Feasible
25	1,000	1,065	Feasible	2,017	Feasible
26	1,000	1,016	Feasible	2,587	Feasible
27	1,000	1,014	Feasible	879	Feasible
28	1,000	1,012	Feasible	724	Feasible
29	1,000	1,303	Feasible	1,405	Feasible
30	1,000	1,025	Feasible	1,520	Feasible

This paper has no intention to make a comprehensive comparison of the proposed method with the existing methods. Nevertheless, one can see from these preliminary numerical results that the numbers of linear programs solved by the method for most instances of two specific problems are less than those of the best CPLEX strategy: branch and cut. An efficient implementation of the method requires a considerable amount of additional research, which is beyond the scope of this paper and will be carried out in another research project.

Appendix: Basis reduction and preconditioning

A subset $L \subset R^n$ is called a lattice if there exist linearly independent vectors $b_1, b_2, \ldots, b_n$ such that $L = \{\sum_{j=1}^{l} \alpha_j b_j \mid \alpha_j$ is an integer for $1 \leq j \leq l\}$. The well-known Gram-Schmidt orthogonalization is a transformation procedure that derives from the independent vectors

$b_j, j = 1, 2, \ldots, l$, the orthogonal vectors $b_j^*, j = 1, 2, \ldots, l$, by the following procedure:

$$b_1^* = b_1,$$
$$b_j^* = b_j - \sum_{k=1}^{j-1} \mu_{jk} b_k^*, \quad 2 \le j \le l,$$
$$\mu_{jk} = \frac{b_j^\top b_k^*}{b_k^{*\top} b_k^*}, \quad 1 \le k < j \le l.$$

For $y \in R^n$, let $\|y\|$ denote the Euclidean norm of y. The following definition comes from [24].

Definition 1 A basis $b_1, b_2, \ldots, b_l$ is called reduced if the following two conditions are satisfied:

Condition 1: $|\mu_{jk}| \le \frac{1}{2}$ for $1 \le k < j \le l$, and
Condition 2: $\|b_j^* + \mu_{j,j-1} b_{j-1}^*\|^2 \ge \frac{3}{4}\|b_{j-1}^*\|^2$ for $1 < j \le l$.

Given this definition, Lováz's basis reduction algorithm in [24] can be stated as follows:

Step 1 (Size reduction): If, for any pair of j and k with $1 \le k < j \le l$, Condition 1 is violated, then replace b_j by $b_j - \lceil \mu_{jk} \rfloor b_k$, where $\lceil \mu_{jk} \rfloor = \lceil \mu_{jk} - \frac{1}{2} \rceil$.
Step 2 (Interchange): If Condition 2 is violated for some j with $1 < j \le l$, then interchange b_{j-1} and b_j.
Step 3 (Repeat): Repeat the above two steps till there is no violation of either of Conditions 1 and 2.

Let $H = \{y \in R_+^m \mid Cy = d\}$, where $C = (c_{ij})$ is an $n \times m$ integer matrix and d is an integer point of R^n. Without loss of generality we assume that $\gcd(c_{i1}, c_{i2}, \ldots, c_{im}) = 1$ for all $1 \le i \le m$. This assumption can be met by directly dividing the GCD (greatest common divisor) to each row of C, where the GCD can be found by an extended GCD algorithm. To convert this problem into an equivalent problem of determining whether there is an integer point in a polytope given by $P = \{x \in R^n \mid Ax \le b\}$, we use the same procedure as in [11]. Let

$$B = \begin{pmatrix} I_m & 0 \\ 0 & N_1 \\ N_2 C & -N_2 d \end{pmatrix},$$

where I_m is an $m \times m$ identity matrix and N_1 and N_2 are two sufficiently large positive integers (*e.g.*, N_1 = 1,000 and N_2 = 10,000). Applying Lováz's basis reduction algorithm to B, we obtain

$$\hat{B} = \begin{pmatrix} A & b & G \\ 0 & N_1 & 0 \\ 0 & 0 & N_2 I_n \end{pmatrix},$$

where I_n is an $n \times n$ identity matrix. Let $P = \{x \in R^n \mid Ax \le b\}$, where A and b are the same as in $\hat{B}$. Then, determining whether there is an integer point in H is equivalent to determining whether there is an integer point in P.

Given any polytope $P = \{x \in R^n \mid Ax \leq b\}$, one can precondition A by applying Lováz's basis reduction algorithm. If there are continuous variables, one can apply Gram-Schmidt orthogonalization to the corresponding matrix.

Given positive integers p_i, $i = 1, 2, \ldots, m$, the extended GCD with the basis reduction [25] can be employed to find their common greatest divisor, which is as follows.

Initialization: Let $m_1 = 3$, $n_1 = 4$, $U = (u_{ij})_{m\times m} = I_m$, $d_i = 1$, $i = 1, 2, \ldots, m+1$, $T = (t_{ij})_{m\times m} = 0_m$, and $k = 2$.

Step 1: Let $i = k-1$ and perform Reduce(k, i). Let

$$r_1 = n_1\bigl(d_{k-1}d_{k+1} + t^2_{k,k-1}\bigr) - m_1 d^2_k.$$

If $p_{k-1} \neq 0$ or $p_{k-1} = 0$, $p_k = 0$, and $r_1 < 0$, then perform Swap(k) and let $k = k-1$ if $k > 2$. Otherwise, for $i = k-2, k-3, \ldots, 1$, perform Reduce(k, i). Let $k = k+1$ and go to *Step* 2.

Step 2: If $k > m$, *Stop*. Otherwise, go to *Step* 1.

Finalization: If $p_m < 0$, let $p_m = -p_m$ and $u_{jm} = -u_{jm}$, $j = 1, 2, \ldots, m$. Let GCD $= p_m$ and, for $j = 1, 2, \ldots, m$, let

$$h = u_{j1},$$

$$u_{j1} = u_{jm},$$

$$u_{jm} = h.$$

Reduce(k, i)

If $p_i \neq 0$, let

$$r = \left\lceil p_k/p_i - \frac{1}{2} \right\rceil.$$

Otherwise, let

$$r = \begin{cases} \lceil t_{ki}/d_{i+1} - \frac{1}{2} \rceil & \text{if } 2|t_{ki}| > d_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

If $r \neq 0$, let

$$p_k = p_k - rp_i,$$

$$u_{jk} = u_{jk} - ru_{ji}, \quad j = 1, 2, \ldots, m,$$

$$t_{ki} = t_{ki} - rd_{i+1},$$

$$t_{kj} = t_{kj} - rt_{ij}, \quad j = 1, 2, \ldots, i-1.$$

Swap(k)

$$h = p_k,$$

$$p_k = p_{k-1},$$

$$p_{k-1} = h.$$

For $j = 1, 2, \dots, m$,

$$h = u_{jk},$$
$$u_{jk} = u_{j,k-1},$$
$$u_{j,k-1} = h.$$

For $j = 1, 2, \dots, k-2$,

$$h = t_{kj},$$
$$t_{kj} = t_{k-1,j},$$
$$t_{k-1,j} = h.$$

For $j = k+1, k+2, \dots, m$,

$$h_0 = t_{j,k-1}t_{k,k-1} + t_{jk}d_{k-1},$$
$$h_1 = t_{j,k-1}d_{k+1} - t_{jk}t_{k,k-1},$$
$$t_{j,k-1} = h_0/d_k,$$
$$t_{jk} = h_1/d_k,$$
$$d_k = \left(d_{k-1}d_{k+1} + t_{k,k-1}^2\right)/d_k.$$

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
Both authors contributed equally to the writing of this paper. Both authors read and approved the final manuscript.

Author details
[1]Department of Systems Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong, China. [2]Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4026, USA.

Acknowledgements
The authors are very grateful to the editor and reviewers for their valuable comments and suggestions and would like to thank Dr. Zhengtian Wu for numerical implementation of the method. This work was partially supported by GRF: CityU 101113 of Hong Kong SAR Government.

Received: 24 June 2015 Accepted: 24 September 2015 Published online: 06 October 2015

References
1. Scarf, HE: Production sets with indivisibilities - Part I: generalities. Econometrica **49**, 1-32 (1981)
2. Scarf, HE: Neighborhood systems for production sets with indivisibilities. Econometrica **54**, 507-532 (1986)
3. Jünger, M, Liebling, TM, Naddef, D, Nemhauser, GL, Pulleyblank, WR, Reinelt, G, Rinaldi, G, Wolsey, LA: 50 Years of Integer Programming. Springer, Berlin (2010)
4. Garey, MR, Johnson, DS: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
5. Gomory, RE: Outline of an algorithm for integer solution to linear programs. Bull. Am. Math. Soc. **64**, 275-278 (1958)
6. Land, AH, Doig, AG: An automatic method for solving discrete programming problems. Econometrica **28**, 497-520 (1960)
7. Lenstra, HW Jr.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**, 538-548 (1983)
8. Lovász, L, Scarf, HE: The generalized basis reduction algorithm. Math. Oper. Res. **17**, 751-764 (1992)
9. Dang, C: An arbitrary starting homotopy-like simplicial algorithm for computing an integer point in a class of polytopes. SIAM J. Discrete Math. **23**, 609-633 (2009)

10. Dang, C, van Maaren, H: A simplicial approach to the determination of an integer point in a simplex. Math. Oper. Res. **23**, 403-415 (1998)
11. Aardal, K, Hurkens, C, Lenstra, AK: Solving a system of linear Diophantine equations with lower and upper bounds on the variables. Math. Oper. Res. **25**, 427-442 (2000)
12. Barvinok, AI: A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. Math. Oper. Res. **19**, 769-779 (1994)
13. Bertsimas, D, Weismantel, R: Optimization over Integers. Dynamic Ideas, Belmont (2005)
14. Burer, S, Vandenbussche, D: A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. Math. Program., Ser. A **113**, 259-282 (2008)
15. Cornuéjols, G, Dawande, M: A class of hard small 0-1 programs. In: Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 1412, pp. 284-293. Springer, Berlin (1998)
16. De Loera, JA, Hemmecke, R, Köppe, M, Weismantel, R: Integer polynomial optimization in fixed dimension. Math. Oper. Res. **31**, 147-153 (2006)
17. Nemhauser, GL, Wolsey, LA: Integer and Combinatorial Optimization. Wiley, New York (1998)
18. Schrijver, A: Theory of Linear and Integer Programming. Wiley, New York (1998)
19. Tarski, A: A lattice-theoretical fixpoint theorem and its applications. Pac. J. Math. **5**, 285-308 (1955)
20. Dang, C, Qi, Q, Ye, Y: Computational models and complexities of Tarski's fixed points. Technical report (2011). http://www.stanford.edu/~yyye/unitarski1.pdf
21. Dang, C: An increasing-mapping approach to integer programming based on lexicographic ordering and linear programming. In: Du, DZ, Zhang, XS (eds.) The Ninth International Symposium on Operations Research and Its Applications. Lecture Notes in Operations Research, vol. 12, pp. 55-60. World Publishing Corporation, Beijing (2010)
22. Ye, Y: Interior Point Algorithms: Theory and Analysis. Wiley, New York (1997)
23. Ye, Y, Todd, MJ, Mizuno, S: An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. Math. Oper. Res. **19**, 53-67 (1994)
24. Lenstra, AK, Lenstra, HW Jr., Lovász, L: Factoring polynomials with rational coefficients. Math. Ann. **261**, 515-534 (1982)
25. Havas, G, Majewski, BS, Matthews, KR: Extended GCD and Hermite normal form algorithms via lattice basis reduction. Exp. Math. **7**, 125-136 (1998)