# Construction of value functions of integer programs with finite domain

Yifei Huang[1], Junlong Zhang[2]

[1]Department of Management, Technology and Economics, ETH Zürich
[2]Department of Industrial Engineering, Tsinghua University

*zhangjunlong@tsinghua.edu.cn*

MIP Workshop 2025, Minneapolis

**Value functions of integer programs**:

$$z(\beta) = \max\{c^T x \mid Ax \le \beta,\ x \in \mathbb{Z}^n_+\},\ \beta \in \mathbb{Z}^m,$$

where $c$ and $A$ are integral. Let $S(\beta) = \{x \in \mathbb{Z}^n_+ \mid Ax \le \beta\}$.

Properties (Nemhauser and Wolsey, 1988):

1. $z(\cdot)$ is **nondecreasing** in $\beta \in \mathbb{Z}^m$.
2. $z(\cdot)$ is **superadditive** over $D = \{\beta \in \mathbb{Z}^m \mid S(\beta) \neq \emptyset\}$. That is, for $\beta^1, \beta^2 \in D$, if $\beta^1 + \beta^2 \in D$, then

$$z(\beta^1) + z(\beta^2) \le z(\beta^1 + \beta^2).$$

# 1. Introduction

$$z(\beta) = \max\{c^T x \mid Ax \le \beta,\ x \in \mathbb{Z}^n_+\},\ \beta \in \mathbb{Z}^m.$$

Properties (Nemhauser and Wolsey, 1988):

3. (**Integer complementary slackness**)
   If $\hat{x} \in \arg\max\{c^T x \mid x \in S(\beta)\}$, then for all $x \in \mathbb{Z}^n_+$ such that $x \le \hat{x}$,

   $$z(Ax) = c^T x$$

   and

   $$z(Ax) + z(\beta - Ax) = z(Ax) + z(A(\hat{x} - x)) = z(\beta).$$

4. (**Column elimination**)
   (1) For $j = 1, \dots, n$, we have $z(a_j) \ge c_j$.
   (2) If $z(a_j) > c_j$, then $\hat{x}_j = 0$ for all $\hat{x} \in \arg\max\{c^T x \mid x \in S(\beta)\}$ and $\beta \in \mathbb{Z}^m$.

# 1. Introduction

Application in solving **stochastic integer programs**:

$$\begin{aligned} \max \quad & c^T x + \mathbb{E}_\xi[Q(x,\xi)] \\ \text{subject to} \quad & Ax \le b,\ x \in \mathbb{Z}_+^{n_1}, \end{aligned}$$

where

$$Q(x,\xi) = \max\{t^T y \mid Wy \le h(\xi) - Tx,\ y \in \mathbb{Z}_+^{n_2}\}.$$

Idea: compute IP value functions in both the first and second stages, and then solve the stochastic integer program.

# 1. Introduction

Application in solving **bilevel integer programs**:

$$\begin{aligned} \max \quad & c^T x + d^T y \\ \text{subject to} \quad & Ax \le b, \; x \in \mathbb{Z}_+^{n_1}, \\ & y \in \arg\max\{t^T y \mid Wy \le h - Tx, \; y \in \mathbb{Z}_+^{n_2}\}. \end{aligned}$$

- Two decision makers: **leader** and **follower**.
- Leader decides **upper-level variables** $x$ first, and follower then decides **lower-level variables** $y$.

Define **follower's value function**:

$$z(\beta) = \max\{t^T y \mid Wy \le \beta, \; y \in \mathbb{Z}_+^{n_2}\}, \; \beta \in \mathbb{Z}^{m_2}.$$

Idea: compute $z(\cdot)$ first, and then solve the bilevel integer program.

# 1. Introduction

Application in solving **multi-objective integer programs**:

$$\begin{aligned} \max \quad & f = (c_1^T x,\ c_2^T x, \ldots, c_p^T x) \\ \text{subject to} \quad & x \in X. \end{aligned}$$

The corresponding value function in $\epsilon$-constraint method:

$$\begin{aligned} z(\beta) = \max_{x \in X} \quad & c_1^T x \\ \text{subject to} \quad & c_2^T x \leq \beta_1, \\ & c_3^T x \leq \beta_2, \\ & \ldots \\ & c_p^T x \leq \beta_{p-1}. \end{aligned}$$

Application in solving other problems:

- Robust optimization problems
- Online optimization problems
- ...

# 1. Introduction

**Construction of IP value functions:**

**Proposition (dynamic programming recursion) (Kong et al. 2006)**

For IP value function $z(\cdot)$ with nonnegative constraint matrix ($A \geq 0$) and objective function coefficients ($c \geq 0$), and a feasible right-hand side vector $\beta \geq a_j$ for some $1 \leq j \leq n$, we have

$$z(\beta) = \max\{z(\beta - a_j) + c_j : \beta \geq a_j,\ 1 \leq j \leq n\}.$$

A dynamic programming algorithm for computing the IP value function $z(\beta)$ **for** $\beta \in [0, \bar{\beta}]$ in:

Kong, N., Schaefer, A. J., & Hunsaker, B. (2006). Two-stage integer programs with stochastic right-hand sides: A superadditive dual approach. *Mathematical Programming*, 108(2-3), 275-296.

*Note: the dimension of $\beta$ is restricted to $m \leq 7$ in their experiments.*

# 1. Introduction

## Definition (level-set minimal) (Trapp et al. 2013)

For IP value function $z(\cdot)$, a right-hand side vector $\beta$ is called *level-set minimal* if $z(\beta - e_i) < z(\beta)$ for all $i \in \{1, \ldots, m\}$, where $e_i$ is the $i$th unit vector. Let $\bar{\mathbf{B}}$ be the set of all level-set minimal vectors.

## Proposition (Trapp et al. 2013)

For any level-set minimal vector $\beta \in \bar{\mathbf{B}}$ and $\hat{x}$ optimal to $z(\beta)$,

$$A\hat{x} = \beta.$$

## Theorem (Trapp et al. 2013)

For any feasible right-hand side vector $\beta$,

$$z(\beta) = \max_{\pi \in \bar{\mathbf{B}} : \pi \leq \beta} z(\pi).$$

# 1. Introduction

A recursive procedure for computing the IP value function $z(\beta)$ for $\beta \in \bar{\mathbf{B}} \cap [0, \bar{\beta}]$ in:

> Trapp, A. C., Prokopyev, O. A., & Schaefer, A. J. (2013). On a level-set characterization of the value function of an integer program and its application to stochastic programming. *Operations Research*, 61(2), 498-511.

Main idea:

1. Solve the IP for some $\beta$, then use integer complementary slackness to derive the optimal solution to some other $\beta$.
2. Generate all $Ax$, then discard those vectors not level-set minimal.

*Note: the dimension of $\beta$ can be larger than that in Kong et al. (2006).*

IP value function

$$z(\beta) = \max_{x}\{c^T x : Ax \leq \beta,\ x \in \mathbb{Z}^n_+\},\ \beta \in \mathbb{Z}^m.$$

Restricted IP value function over the first $k \in \{1, \ldots, n\}$ variables

$$z_k(\beta) = \max_{x}\left\{\sum_{j=1}^{k} c_j x_j : \sum_{j=1}^{k} a_j x_j \leq \beta,\ x \in \mathbb{Z}^k_+\right\},\ \beta \in \mathbb{Z}^m.$$

was defined in:

> Brown S, Zhang W, Ajayi T, Schaefer AJ (2021). A Gilmore-Gomory construction of integer programming value functions. *Operations Research Letters*, 49(4):522–529.

Define $S_k(\beta) = \{x \in \mathbb{Z}^k_+ : \sum_{j=1}^{k} a_j x_j \leq \beta\}$, $\mathbf{B}_k = \{\beta \in \mathbb{Z}^m : S_k(\beta) \neq \emptyset\}$ and $\text{opt}_k(\beta) = \arg\max\{\sum_{j=1}^{k} c_j x_j : x \in S_k(\beta)\}$. Let $\bar{\mathbf{B}}_k$ be the set of all level-set minimal vectors in $\mathbf{B}_k$.

# 2. A column-by-column construction approach

## Proposition (Brown et al. 2021)

For $k \in \{2, \dots, n\}$, we have

$$\bar{\mathbf{B}}_k \subseteq \left\{ \hat{\beta} + ta_k : \hat{\beta} \in \bar{\mathbf{B}}_{k-1},\ t \in \mathbb{Z}_+ \right\}.$$

## Lemma (Initial condition)

$z_1(ta_1) = tc_1$ for $t \geq 1$ and $\bar{\mathbf{B}}_1 = \{x_1 a_1 : x_1 \in \mathbb{Z}_+\}$.

## Lemma (Column elimination)

For $2 \leq k \leq \hat{k} \leq n$,

(i) for $t \geq 1$, if $z_{\hat{k}}(ta_k) > tc_k$ or $ta_k \notin \bar{\mathbf{B}}_{\hat{k}}$, then $ta_k$ can be discarded in the generation of $\bar{\mathbf{B}}_{\hat{k}}$;

(ii) for $1 \leq j < k \leq \hat{k} \leq n$ and $t \geq 1$, if $z_j(ta_k) = tc_k$, then for all $\beta \in \mathbf{B}_{\hat{k}}$ with $\beta \geq ta_k$, there exists $x^* \in \mathrm{opt}_{\hat{k}}(\beta)$ such that $x^*_k < t$.

**Proposition**

For $k \in \{2, \ldots, n\}$ and $t \geq 2$, suppose that $z_k((t-1)a_k) = (t-1)c_k$. We have

(i) if $z_{k-1}(ta_k) \leq tc_k$, then $z_k(ta_k) = tc_k$;

(ii) if $z_{k-1}(ta_k) < tc_k$ and $(t-1)a_k \in \bar{\mathbf{B}}_k$, then $ta_k \in \bar{\mathbf{B}}_k$;

(iii) if $z_{k-1}(ta_k) > tc_k$, then $z_k(ta_k) = z_{k-1}(ta_k)$.

Thus, $z_k(ta_k) = \max\{z_{k-1}(ta_k),\ tc_k\}$.

The following proposition implies that we can discard those $\beta \in \bar{\mathbf{B}}_{k-1} \backslash \bar{\mathbf{B}}_k$ in the construction of $\bar{\mathbf{B}}_k$.

**Proposition**

For $k \in \{2, \dots, n\}$ and $\beta \in \bar{\mathbf{B}}_k \backslash \bar{\mathbf{B}}_{k-1}$, there exists $\hat{\beta} \in \bar{\mathbf{B}}_{k-1} \cap \bar{\mathbf{B}}_k$ and $t \geq 1$ such that $\beta = \hat{\beta} + ta_k$.

The following lemma is critical in deriving the rest properties.

**Lemma**

For $k \in \{2, \dots, n\}$ and $t \geq 0$, we have

$$z_k(\beta + ta_k) = \max\{z_{k-1}(\beta + ta_k),\ z_k(\beta + (t-1)a_k) + c_k\}.$$

# 2. A column-by-column construction approach

The following two propositions present sufficient conditions under which $\beta + ta_k$ is NOT level-set minimal with respect to $z_k(\cdot)$.

**Proposition**

For $k \in \{2, \ldots, n\}$, $t \geq 0$ and $\beta + ta_k \in \mathbf{B}_{k-1}$, if $\beta + ta_k \notin \bar{\mathbf{B}}_{k-1}$ and $z_{k-1}(\beta + ta_k) \geq z_k(\beta + (t-1)a_k) + c_k$, then $\beta + ta_k \notin \bar{\mathbf{B}}_k$.

**Proposition**

For $k \in \{2, \ldots, n\}$, $t \geq 0$ and $\beta + ta_k \in \mathbf{B}_{k-1}$, if $\beta + (t-1)a_k \notin \bar{\mathbf{B}}_k$ and $z_{k-1}(\beta + ta_k) \leq z_k(\beta + (t-1)a_k) + c_k$, then $\beta + ta_k \notin \bar{\mathbf{B}}_k$.

# 2. A column-by-column construction approach

The following two propositions present sufficient conditions under which $\beta + ta_k$ is level-set minimal with respect to $z_k(\cdot)$.

**Proposition**

For $k \in \{2, \dots, n\}$, $t \geq 0$ and $\beta + ta_k \in \mathbf{B}_{k-1}$, if $\beta + (t-1)a_k \in \bar{\mathbf{B}}_k$ and $z_{k-1}(\beta + ta_k) < z_k(\beta + (t-1)a_k) + c_k$, then $\beta + ta_k \in \bar{\mathbf{B}}_k$.

**Proposition**

For $k \in \{2, \dots, n\}$, $t \geq 0$ and $\beta + ta_k \in \bar{\mathbf{B}}_{k-1}$, if $z_{k-1}(\beta + ta_k) > z_k(\beta + (t-1)a_k) + c_k$, or $\beta + (t-1)a_k \in \bar{\mathbf{B}}_k$ and $z_{k-1}(\beta + ta_k) \geq z_k(\beta + (t-1)a_k) + c_k$, then $\beta + ta_k \in \bar{\mathbf{B}}_k$.

# 2. A column-by-column construction approach

The following proposition gives a sufficient condition under which $\beta + (t+1)a_k$ does not need to be considered in the construction of $\bar{\mathbf{B}}_k$.

## Proposition

For $k \in \{2, \ldots, n\}$, $t \geq 1$ and $\beta \in \mathbf{B}_{k-1}$, if $\beta \in \bar{\mathbf{B}}_k$ and $\beta + ta_k \notin \bar{\mathbf{B}}_k$, then $\beta + (t+1)a_k \notin \bar{\mathbf{B}}_k \backslash \bar{\mathbf{B}}_{k-1}$.

**Algorithm 1** Construct $z_n(\cdot)$ over $\bar{\mathrm{B}}_n \cap \mathcal{H}_+$ when $a_k \gneq 0$ for all $k \in \{1, \ldots, n\}$

**Initialization:** Compute $u_k = \min\{\lfloor \frac{\bar{h}_i}{a_{ik}} \rfloor : i = 1, \ldots, m\}$ for each $k \in \{1, \ldots, n\}$. Set $z_1(ta_1) = tc_1$ for $t = 0, \ldots, u_1$, and $\bar{\mathcal{B}}_1 = \bar{\mathrm{B}}_1 \cap \mathcal{H}_+ = \{ta_1 : t = 0, \ldots, u_1\}$ (Lemma 3). Initialize $k^- = 1$ and $k = 2$.

```
1:  while k ≤ n do
2:      Initialize B̄_k = ∅.
3:      for t = 1 to u_k do
4:          Compute z_{k−}(ta_k) = max{z_{k−}(β̃) : β̃ ∈ B̄_{k−}, β̃ ≤ ta_k}. (Proposition 3)
5:          if z_{k−}(ta_k) ≥ tc_k then
6:              if t == 1 then
7:                  Eliminate a_k, set k = k + 1 and go to line 1. (Propositions 4 and 10)
8:              else
9:                  Go to line 11. (Propositions 9 and 10)
10:         Set z_k(ta_k) = tc_k and B̄_k = B̄_k ∪ {ta_k}. (Propositions 11 and 14)
11:     Update u_k = t − 1.
```

# 2. A column-by-column construction approach

12: **for all** $\beta \in \bar{\mathcal{B}}_{k^-}$ (in lexicographical order) **do**

13: **if** $\beta \not\geq a_k$ **then**

14: Set $\bar{\mathcal{B}}_k = \bar{\mathcal{B}}_k \cup \{\beta\}$ and $z_k(\beta) = z_{k^-}(\beta)$. (Proposition 16)

15: **else**

16: Compute $z_k(\beta - a_k) = \max\{z_k(\tilde{\beta}) : \tilde{\beta} \in \bar{\mathcal{B}}_k,\ \tilde{\beta} \leq \beta - a_k\}$.

17: Set $z_k(\beta) = \max\{z_{k^-}(\beta),\ z_k(\beta - a_k) + c_k\}$. (Lemma 6)

18: **if** $z_{k^-}(\beta) > z_k(\beta - a_k) + c_k$ or $\beta - a_k \in \bar{\mathcal{B}}_k$ **then**

19: Set $\bar{\mathcal{B}}_k = \bar{\mathcal{B}}_k \cup \{\beta\}$. (Propositions 20 and 21)

20: **else**

21: $\beta \notin \bar{\mathbf{B}}_k$ and go to line 12. (Propositions 17 and 19)

22: **for** $t = 1$ to $u_k$ **do**

23: **if** $\beta + ta_k \not\leq \bar{h}$ **then** Go to line 12.

24: Compute $z_{k^-}(\beta + ta_k) = \max\{z_{k^-}(\tilde{\beta}) : \tilde{\beta} \in \bar{\mathcal{B}}_{k^-},\ \tilde{\beta} \leq \beta + ta_k\}$.

25: Set $z_k(\beta + ta_k) = \max\{z_{k^-}(\beta + ta_k),\ z_k(\beta + (t-1)a_k) + c_k\}$. (Lemma 6)

26: **if** $z_{k^-}(\beta + ta_k) < z_k(\beta + (t-1)a_k) + c_k$ or $\beta + ta_k \in \bar{\mathcal{B}}_{k^-}$ **then**

27: Set $\bar{\mathcal{B}}_k = \bar{\mathcal{B}}_k \cup \{\beta + ta_k\}$. (Propositions 20 and 21)

28: **else**

29: $\beta + ta_k \notin \bar{\mathbf{B}}_k$ and go to line 12. (Propositions 18 and 22)

30: Set $k^- = k$ and $k = k + 1$.

31: **return** $\bar{\mathcal{B}}_{k^-} = \bar{\mathbf{B}}_n \cap \mathcal{H}_+$ with $z_{k^-}(\beta) = z_n(\beta)$ for $\beta \in \bar{\mathcal{B}}_{k^-}$.

# 2. A column-by-column construction approach

Advantages over the approach of Trapp et al. (2013):

1. Do not need to solve any IP.
2. Do not generate any vector not level-set minimal.
3. Pre-processing is less complicated.
4. Can be extended to the case where $A$ has negative elements.

Table: Characteristics of instance classes in Trapp et al. (2013).

| Class | $\|\Omega\|$ | $m$ | $n_1$ | $n_2$ | $\Delta$ | Range: [min, max] | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $c$ | $T$ | $d$ | $W$ | $h(\omega)$ |
| IC-K1 | 7,812 | 6 | 1,000 | 100 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 9] |
| IC-K2 | 7,812 | 6 | 500 | 500 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 9] |
| IC-K3 | 7,812 | 6 | 300 | 500 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 9] |
| IC-K4 | 7,812 | 6 | 500 | 500 | 0.5 | [1, 10] | [1, 4] | [2, 6] | [2, 5] | [5, 9] |
| IC-K5 | 7,812 | 6 | 500 | 500 | 0.5 | [1, 5] | [1, 3] | [2, 6] | [2, 5] | [5, 9] |
| IC-K6 | 7,812 | 6 | 500 | 500 | 0.8 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 9] |
| IC-K7 | 7,812 | 7 | 1,000 | 500 | 0.7 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-K8 | 7,812 | 7 | 500 | 500 | 0.7 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-K9 | 7,812 | 7 | 300 | 500 | 0.7 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-K10 | 7,812 | 7 | 1,000 | 500 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T1 | 1,000 | 20 | 50 | 50 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T2 | 1,000 | 20 | 50 | 50 | 0.4 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T3 | 1,000 | 20 | 50 | 50 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T4 | 5,000 | 50 | 100 | 100 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T5 | 5,000 | 50 | 100 | 100 | 0.4 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T6 | 5,000 | 50 | 100 | 100 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T7 | 8,000 | 100 | 200 | 200 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T8 | 8,000 | 100 | 200 | 200 | 0.4 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T9 | 8,000 | 100 | 200 | 200 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T10 | 10,000 | 100 | 300 | 300 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T11 | 10,000 | 100 | 300 | 300 | 0.4 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T12 | 10,000 | 100 | 300 | 300 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T13 | 10,000 | 20 | 50 | 50 | 0.5 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T14 | 50,000 | 50 | 100 | 100 | 0.4 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T15 | 80,000 | 100 | 200 | 200 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |
| IC-T16 | 300,000 | 250 | 1,000 | 500 | 0.3 | [1, 5] | [1, 4] | [2, 6] | [2, 5] | [5, 10] |

## 2. A column-by-column construction approach

| Class | Our algorithm | | | Procedure in Trapp et al. (2013) | | | | | $t_P/t_{A1}$ |
|---|---|---|---|---|---|---|---|---|---|
| | $\lvert\bar{\mathbf{B}}^1\rvert$ | $\lvert\bar{\mathbf{B}}^2\rvert$ | $t_{A1}$ | IPs | $t_{CE}$ | $t_{VFC}$ | $t_{RDM}$ | $t_P$ | |
| IC-K1 | 46,656 | 225 | 20.694 | 1,175 | 4.664 | 40.066 | 12.395 | 57.151 | 2.762 |
| IC-K2 | 15,552 | 12,920 | 1.241 | 101,726 | 4.203 | 297.989 | 2.623 | 304.840 | 245.641 |
| IC-K3 | 6,480 | 6,480 | 0.088 | 75,569 | 3.371 | 192.231 | 0.482 | 196.108 | 2,228.500 |
| IC-K4 | 15,552 | 12,920 | 1.250 | 101,726 | 4.281 | 281.205 | 2.568 | 288.077 | 230.462 |
| IC-K5 | 31,104 | 8,064 | 4.378 | 791,683 | 4.373 | 1,972.205 | 5.398 | 1,982.004 | 452.719 |
| IC-K6 | 2,258 | 2,993 | 0.170 | 37,615 | 5.421 | 316.061 | 0.100 | 321.603 | 1,891.782 |
| IC-K7 | 24,378 | 7,685 | 4.794 | 176,298 | 7.827 | 1,049.686 | 4.688 | 1,062.230 | 221.575 |
| IC-K8 | 10,549 | 15,399 | 5.219 | 377,312 | 5.581 | 4,154.700 | 2.610 | 4,162.910 | 797.645 |
| IC-K9 | 5,781 | 3,644 | 0.420 | 49,379 | 4.054 | 338.047 | 0.349 | 342.473 | 815.412 |
| IC-K10 | 279,936 | 28,000 | 98.680 | - | - | - | - | - | - |
| IC-T1 | 18,860 | 42,045 | 8.666 | 32,061 | 1.235 | 95.705 | 18.708 | 115.673 | 13.348 |
| IC-T2 | 1,037 | 1,407 | 0.022 | 2,215 | 1.710 | 2.780 | 0.020 | 4.533 | 202.985 |
| IC-T3 | 227 | 412 | 0.005 | 1,498 | 2.083 | 0.765 | 0.001 | 2.871 | 574.133 |
| IC-T4 | 2,525 | 3,846 | 0.184 | 6,571 | 5.916 | 6.287 | 0.165 | 12.392 | 67.468 |
| IC-T5 | 338 | 450 | 0.019 | 4,744 | 7.737 | 1.064 | 0.002 | 8.826 | 472.821 |
| IC-T6 | 139 | 190 | 0.009 | 5,466 | 9.729 | 0.494 | 0.000 | 10.246 | 1,182.192 |
| IC-T7 | 886 | 988 | 0.118 | 13,697 | 26.288 | 3.096 | 0.013 | 29.423 | 249.350 |
| IC-T8 | 250 | 270 | 0.039 | 16,795 | 36.772 | 1.057 | 0.001 | 37.855 | 970.641 |
| IC-T9 | 201 | 201 | 0.032 | 20,754 | 48.341 | 1.063 | 0.001 | 49.429 | 1,560.926 |
| IC-T10 | 2,094 | 4,043 | 0.579 | 22,751 | 44.231 | 11.321 | 0.164 | 55.743 | 96.329 |
| IC-T11 | 389 | 457 | 0.091 | 25,248 | 63.005 | 1.985 | 0.003 | 65.017 | 711.869 |
| IC-T12 | 302 | 306 | 0.070 | 31,111 | 86.335 | 1.934 | 0.001 | 88.296 | 1,261.367 |
| IC-T13 | 280 | 360 | 0.005 | 1,472 | 2.055 | 0.778 | 0.002 | 2.856 | 535.500 |
| IC-T14 | 279 | 496 | 0.017 | 4,730 | 7.946 | 1.017 | 0.002 | 8.987 | 518.500 |
| IC-T15 | 1,057 | 1,046 | 0.127 | 13,875 | 26.145 | 3.468 | 0.016 | 29.656 | 234.126 |
| IC-T16 | 1,094 | 537 | 0.958 | 115,408 | 772.078 | 12.064 | 0.013 | 784.191 | 818.287 |
| Average | 18,008 | 5,976 | 5.687 | 81,235 | 47.415 | 351.483 | 2.013 | 400.936 | 654.254 |

- Runtime limit is exceeded.

1. The size of set $\bar{\mathbf{B}}$ can be very large. We are working on tighter characterizations of integer programming value functions.
2. We are also studying value functions of binary programs.
3. Future research: develop machine learning approaches for computing integer programming value functions, e.g.,

Bertsimas D, Stellato B (2022). Online mixed-integer optimization in milliseconds. *INFORMS Journal on Computing*, 34(4):2229–2248.

*Thank you!*