(Near-)Optimal Algorithms for Sparse Separable Convex Integer Programs

Christoph Hunkenschröder[1], Martin Koutecký[3], Asaf Levin[2], Tung Anh Vu[3]

[1]TU Berlin

[2]Technion – Israel Institute of Technology

[3]Charles University

IP and ILP

Integer (Linear) Programming problem in standard form:

$$\min\left\{\mathbf{w}^{\mathsf{T}}\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\right\}, \text{ and} \qquad \text{(ILP)}$$

$$\min\left\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\right\} \qquad \text{(IP)}$$

with

- $A \in \mathbb{Z}^{m \times n}$,
- $\mathbf{l}, \mathbf{u}, \mathbf{w} \in \mathbb{Z}^n$, and
- $f : \mathbb{R}^n \to \mathbb{R}$ a separable convex function, i.e. f is expressible as $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ with each $f_i : \mathbb{R} \to \mathbb{R}$ convex.

IP and ILP

Integer (Linear) Programming problem in standard form:

$$\min\left\{\mathbf{w}^{\mathsf{T}}\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\right\}, \text{ and} \qquad \text{(ILP)}$$

$$\min\left\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\right\} \qquad \text{(IP)}$$

with

- $A \in \mathbb{Z}^{m \times n}$,
- $\mathbf{l}, \mathbf{u}, \mathbf{w} \in \mathbb{Z}^n$, and
- $f : \mathbb{R}^n \to \mathbb{R}$ a separable convex function, i.e. f is expressible as $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ with each $f_i : \mathbb{R} \to \mathbb{R}$ convex.

ILP is already NP-hard

IP and ILP

Integer (Linear) Programming problem in standard form:

$$\min\left\{\mathbf{w}^{\mathsf{T}}\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\right\}, \text{ and} \qquad \text{(ILP)}$$

$$\min\left\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b},\, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\, \mathbf{x} \in \mathbb{Z}^n\right\} \qquad \text{(IP)}$$

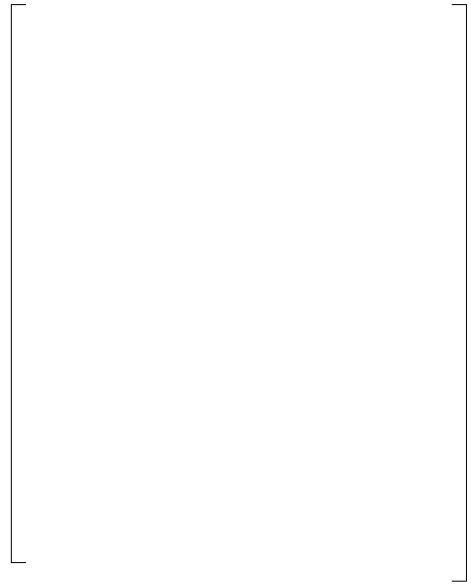
with

- $A \in \mathbb{Z}^{m \times n}$,
- $\mathbf{l}, \mathbf{u}, \mathbf{w} \in \mathbb{Z}^n$, and
- $f : \mathbb{R}^n \to \mathbb{R}$ a separable convex function, i.e. f is expressible as $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ with each $f_i : \mathbb{R} \to \mathbb{R}$ convex.

ILP is already NP-hard $\Rightarrow$ focus on tractable subclasses corresponding to block-structured matrices.

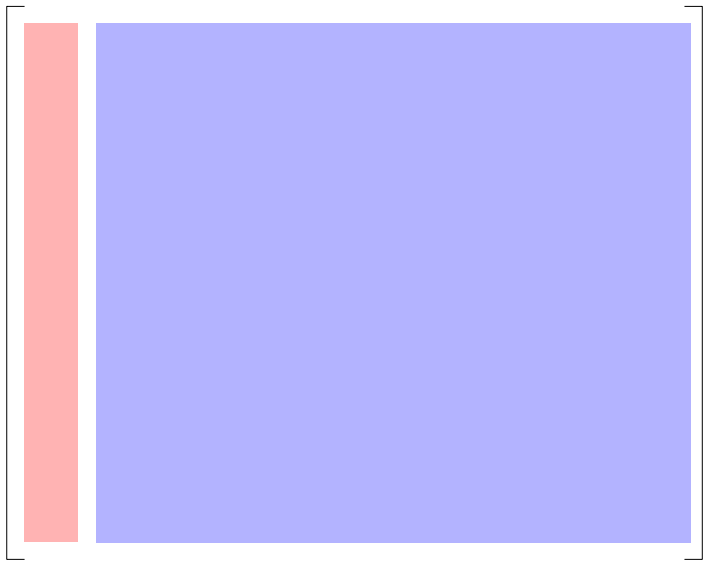
Block Structured Matrices

Multistage-stochastic matrix.

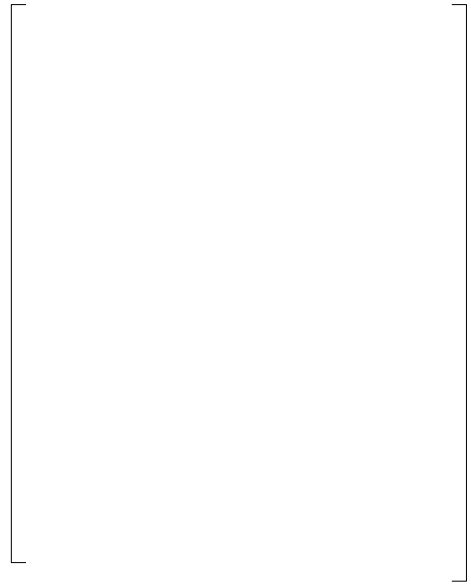


Tree-fold matrix.

Block Structured Matrices

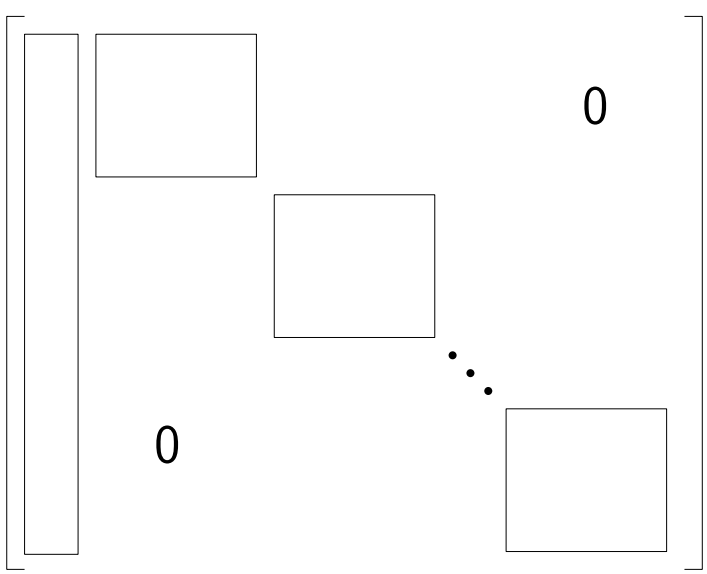


Multistage-stochastic matrix.

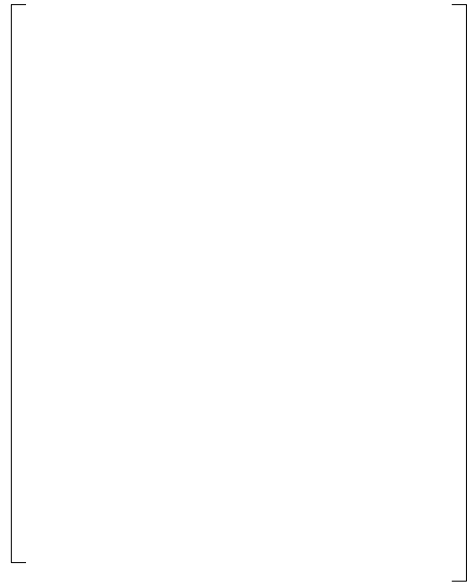


Tree-fold matrix.

Block Structured Matrices

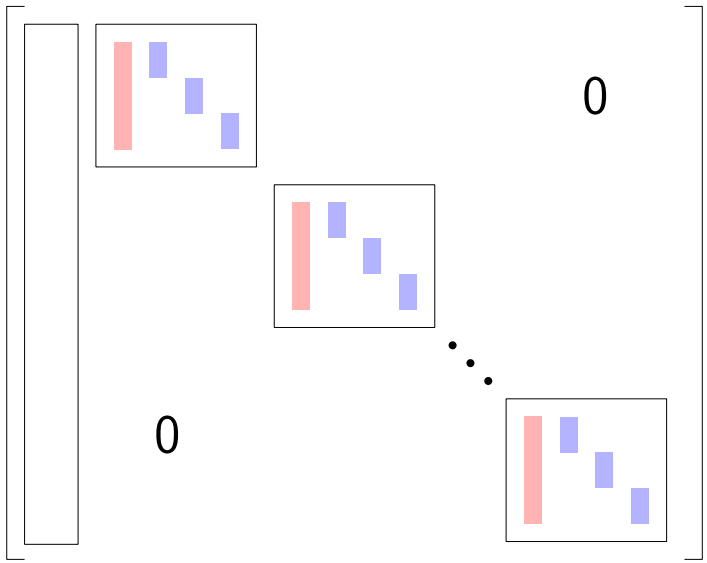


Multistage-stochastic matrix.

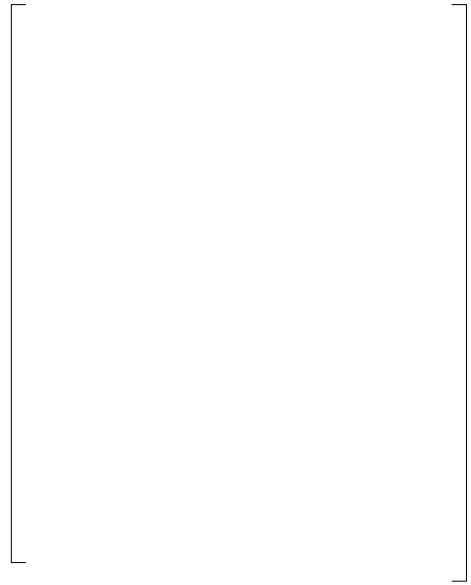


Tree-fold matrix.

Block Structured Matrices

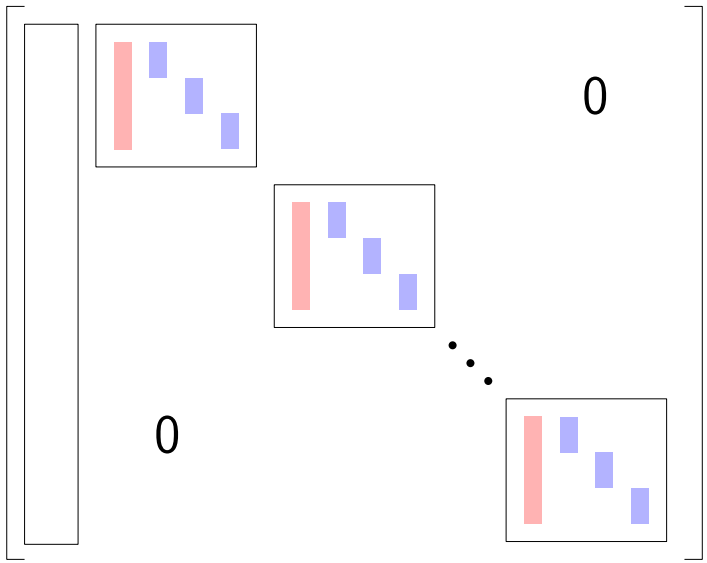


Multistage-stochastic matrix.

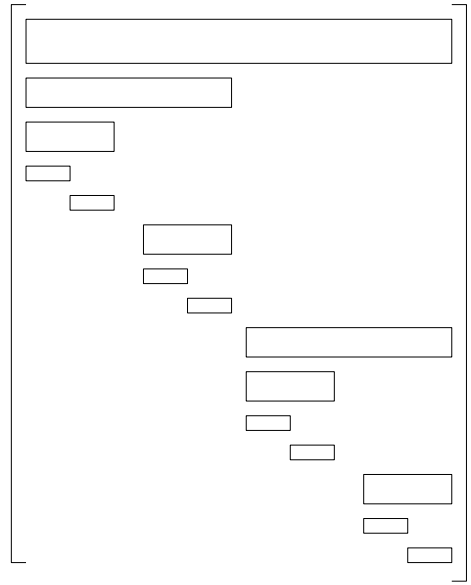


Tree-fold matrix.

Block Structured Matrices

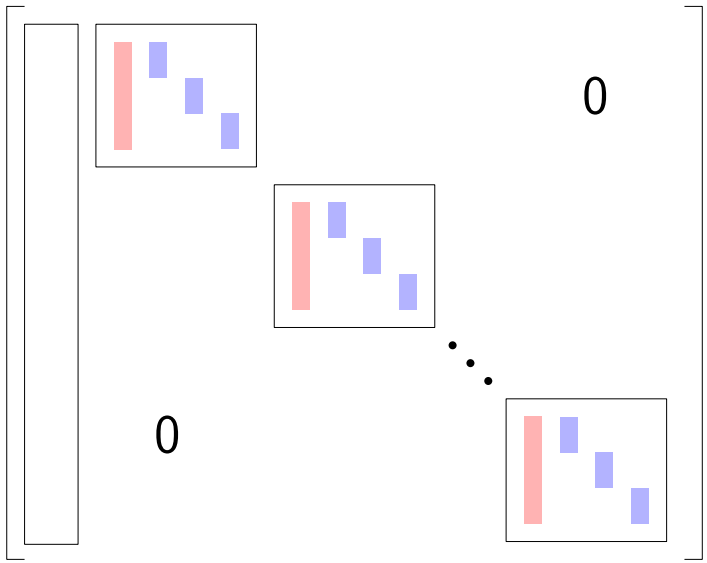


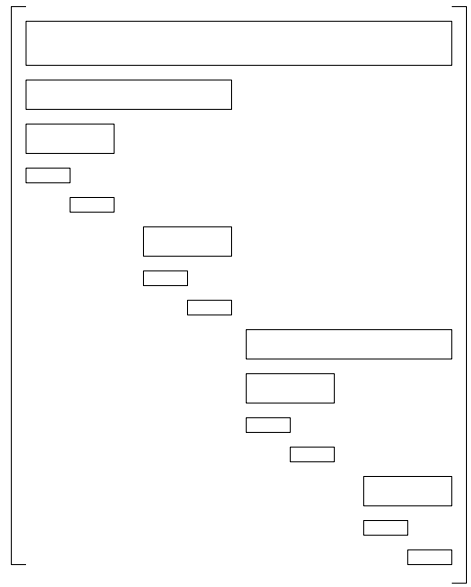
Multistage-stochastic matrix.



Tree-fold matrix.

Block Structured Matrices





Multistage-stochastic matrix.

$\Updownarrow$

Small primal treedepth $\mathrm{td}_P(A)$.

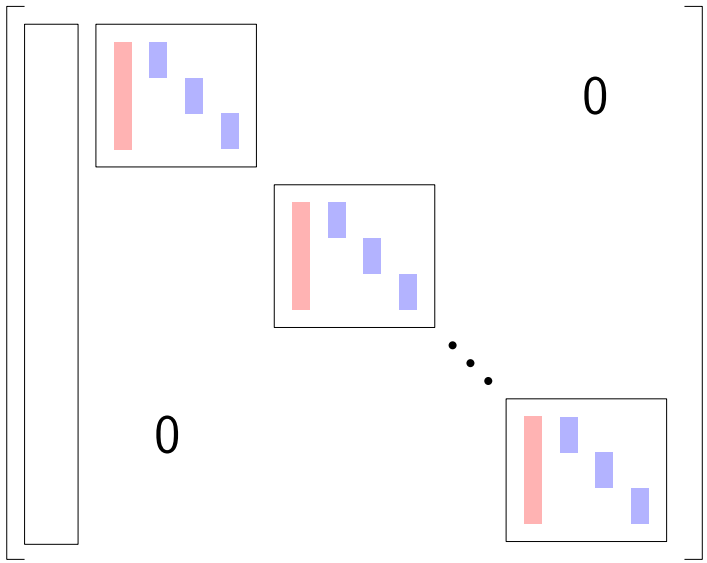
Tree-fold matrix.

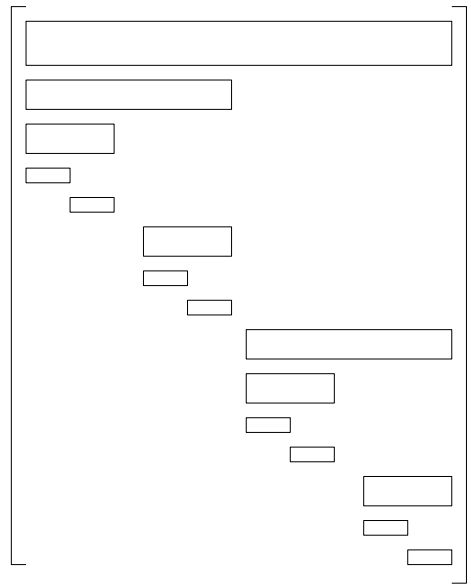
$\Updownarrow$

Small dual treedepth $\mathrm{td}_D(A)$.

- Treedepth measures the similarity of a graph to a star.

Block Structured Matrices





Multistage-stochastic matrix.

$\Updownarrow$

Small primal treedepth $\mathrm{td}_P(A)$.

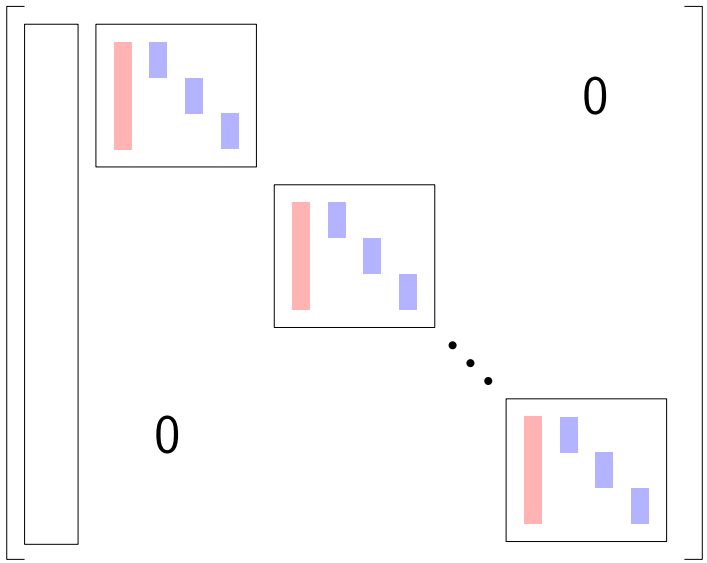
Tree-fold matrix.

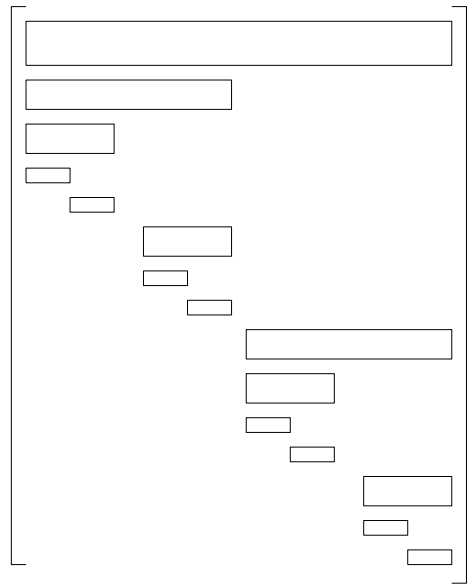
$\Updownarrow$

Small dual treedepth $\mathrm{td}_D(A)$.

- Treedepth measures the similarity of a graph to a star.
- $\mathrm{td}_P(A)$: treedepth of the primal graph of A.

Block Structured Matrices





Multistage-stochastic matrix.

$\Updownarrow$

Small primal treedepth $\mathrm{td}_P(A)$.

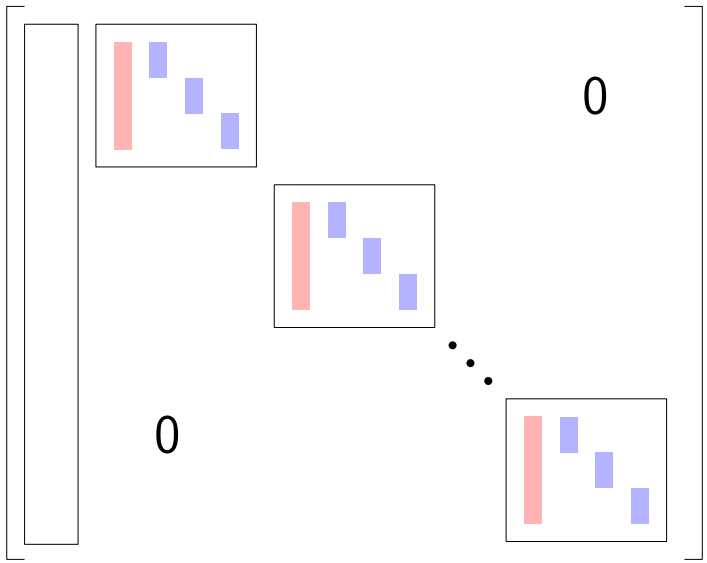
Tree-fold matrix.

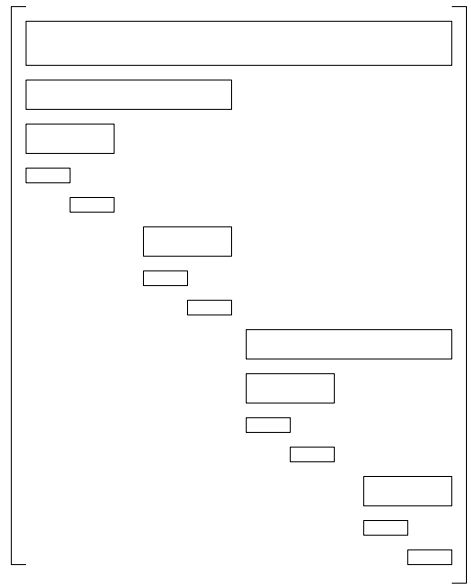
$\Updownarrow$

Small dual treedepth $\mathrm{td}_D(A)$.

- Treedepth measures the similarity of a graph to a star.
- $\mathrm{td}_P(A)$: treedepth of the primal graph of A.
- $\mathrm{td}_D(A)$: treedepth of the primal graph of $A^\intercal$.

Block Structured Matrices





Multistage-stochastic matrix.

$\Updownarrow$

Small primal treedepth $\mathrm{td}_P(A)$.

Tree-fold matrix.

$\Updownarrow$

Small dual treedepth $\mathrm{td}_D(A)$.

- Treedepth measures the similarity of a graph to a star.
- $\mathrm{td}_P(A)$: treedepth of the primal graph of A.
- $\mathrm{td}_D(A)$: treedepth of the primal graph of $A^\intercal$.

Main point. Block structure-ness of $A \approx$ treedepth.

State of the Art

$g, g', g'', \ldots$: some computable function.

State of the Art

$g, g', g'', \ldots$: some computable function.

[Eisenbrand, Hunkenschröder, Klein, Kouteckỳ, Levin, Onn; MOR 2024]

IP can be solved in time

$$g(\|A\|_\infty, d) \; (n^\omega + \min(m,n)nm) \; \log \|\mathbf{u} - \mathbf{l}\|_\infty \log f_{\text{gap}} \;,$$

where $d = \min\{\text{td}_P(A), \text{td}_D(A)\}$, $f_{\text{gap}} = \max_{\mathbf{x},\mathbf{y}:\, \mathbf{l} \le \mathbf{x},\mathbf{y} \le \mathbf{u}} f(\mathbf{x}) - f(\mathbf{y})$.

State of the Art

$g, g', g'', \ldots$: some computable function.

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024]

IP can be solved in time

$$g(\|A\|_\infty, d) \; (n^\omega + \min(m,n)nm) \; \log \|\mathbf{u} - \mathbf{l}\|_\infty \log f_{\text{gap}} \; ,$$

where $d = \min\{\text{td}_P(A), \text{td}_D(A)\}$, $f_{\text{gap}} = \max_{\mathbf{x},\mathbf{y}:\, \mathbf{l} \le \mathbf{x},\mathbf{y} \le \mathbf{u}} f(\mathbf{x}) - f(\mathbf{y})$.

[Cslovjecsek, Eisenbrand, Hunkenschröder, Rohwedder, Weismantel; SODA 2021] [Cslovjecsek, Eisenbrand, Pilipczuk, Venzin, Weismantel; ESA 2021]

ILP can be solved in strongly near-linear time of

$$g(\|A\|_\infty, d) \; n \log^{2^{\mathcal{O}(d)}} n \; .$$

State of the Art

$g, g', g'', \ldots$: some computable function.

[Eisenbrand, Hunkenschröder, Klein, Kouteckỳ, Levin, Onn; MOR 2024]

IP can be solved in time

$$g(\|A\|_\infty, d) \; (n^\omega + \min(m,n)nm) \; \log \|\mathbf{u} - \mathbf{l}\|_\infty \log f_{\text{gap}} \; ,$$

where $d = \min\{\text{td}_P(A), \text{td}_D(A)\}$, $f_{\text{gap}} = \max_{\mathbf{x},\mathbf{y}:\, \mathbf{l} \le \mathbf{x},\mathbf{y} \le \mathbf{u}} f(\mathbf{x}) - f(\mathbf{y})$.

[Cslovjecsek, Eisenbrand, Hunkenschröder, Rohwedder, Weismantel; SODA 2021] [Cslovjecsek, Eisenbrand, Pilipczuk, Venzin, Weismantel; ESA 2021]

ILP can be solved in strongly near-linear time of

$$g(\|A\|_\infty, d) \; n \log^{2^{\mathcal{O}(d)}} n \; .$$

Our motivation

Is there a strongly near-linear time algorithm for IP?

State of the Art

$g, g', g'', \ldots$: some computable function.

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024]

IP can be solved in time

$$g(\|A\|_\infty, d) \; (n^\omega + \min(m,n)nm) \; \log \|\mathbf{u} - \mathbf{l}\|_\infty \log f_{\text{gap}} \,,$$

where $d = \min\{\text{td}_P(A), \text{td}_D(A)\}$, $f_{\text{gap}} = \max_{\mathbf{x},\mathbf{y}:\, \mathbf{l} \le \mathbf{x},\mathbf{y} \le \mathbf{u}} f(\mathbf{x}) - f(\mathbf{y})$.

[Cslovjecsek, Eisenbrand, Hunkenschröder, Rohwedder, Weismantel; SODA 2021] [Cslovjecsek, Eisenbrand, Pilipczuk, Venzin, Weismantel; ESA 2021]

ILP can be solved in strongly near-linear time of

$$g(\|A\|_\infty, d) \; n \log^{2^{\mathcal{O}(d)}} n \,.$$

Our motivation

Is there a strongly near-linear time algorithm for IP? **No!**

$\min f = \sum f_i \Leftrightarrow \min f_i$ separately $\Leftarrow \Omega(\log(u_i - l_i))$ comparisons by information theory.

Our Results: Matching the Lower Bound

Theorem 1

There is an algorithm which solves IP in time

$$g(\mathrm{td}_P(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty .$$

Our Results: Matching the Lower Bound

Theorem 1

There is an algorithm which solves IP in time

$$g(\mathrm{td}_P(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty .$$

Theorem 2

There is an algorithm which solves IP in time

$$g(\mathrm{td}_D(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty \log n .$$

Our Results: Matching the Lower Bound

Theorem 1

There is an algorithm which solves IP in time

$$g(\mathrm{td}_P(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty .$$

Theorem 2

There is an algorithm which solves IP in time

$$g(\mathrm{td}_D(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty \log n .$$

- $\log \|\mathbf{b}\|_\infty \Leftarrow$ finding an initial feasible solution.

Our Results: Matching the Lower Bound

Theorem 1

There is an algorithm which solves IP in time

$$g(\mathrm{td}_P(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty .$$

Theorem 2

There is an algorithm which solves IP in time

$$g(\mathrm{td}_D(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty \log n .$$

- $\log \|\mathbf{b}\|_\infty \Leftarrow$ finding an initial feasible solution.
- **Conjecture:** The dependence on n in Theorem 2 is optimal.

Our Results: Matching the Lower Bound

Theorem 1

There is an algorithm which solves IP in time

$$g(\mathrm{td}_P(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty .$$

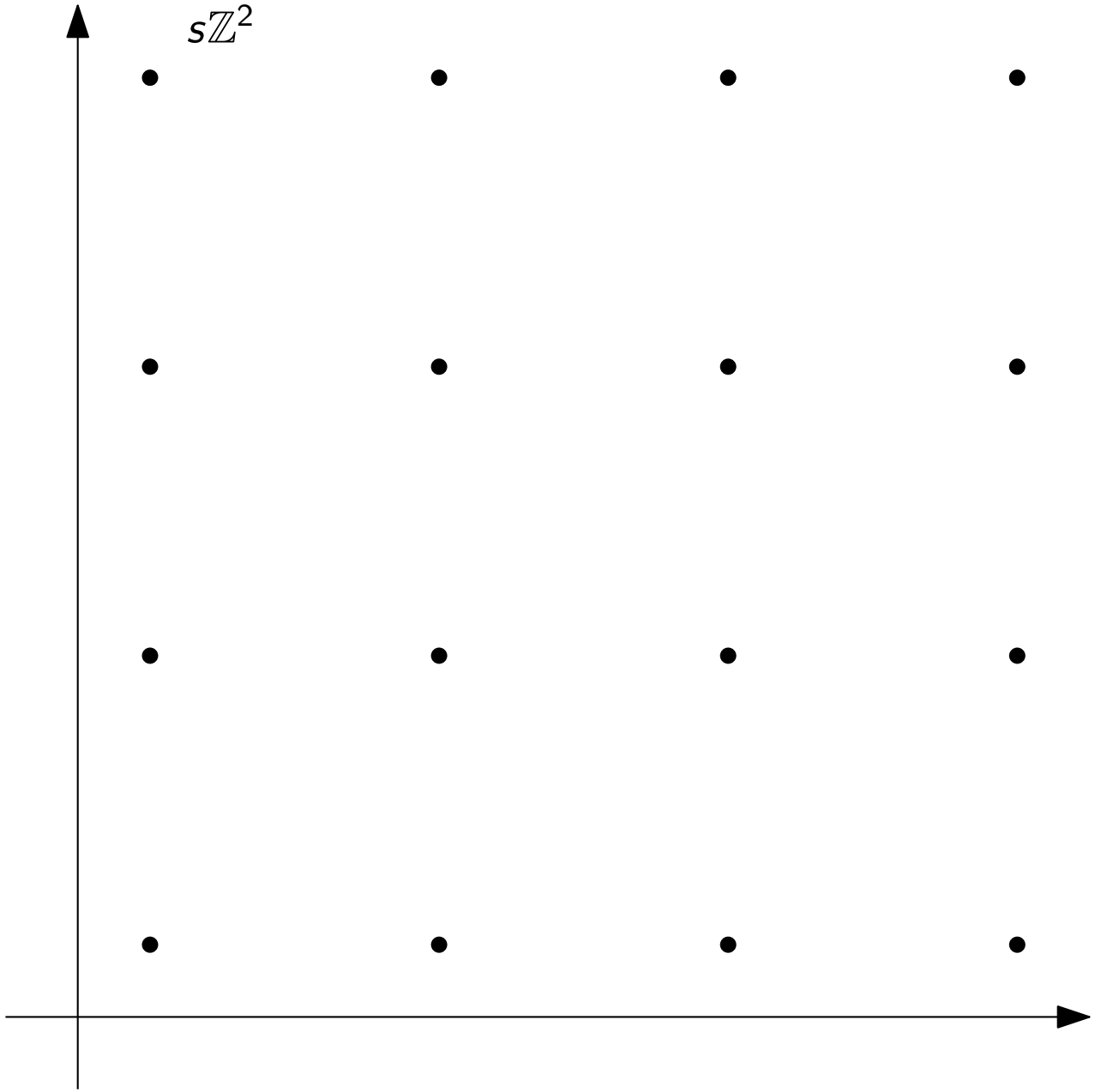
Theorem 2

There is an algorithm which solves IP in time

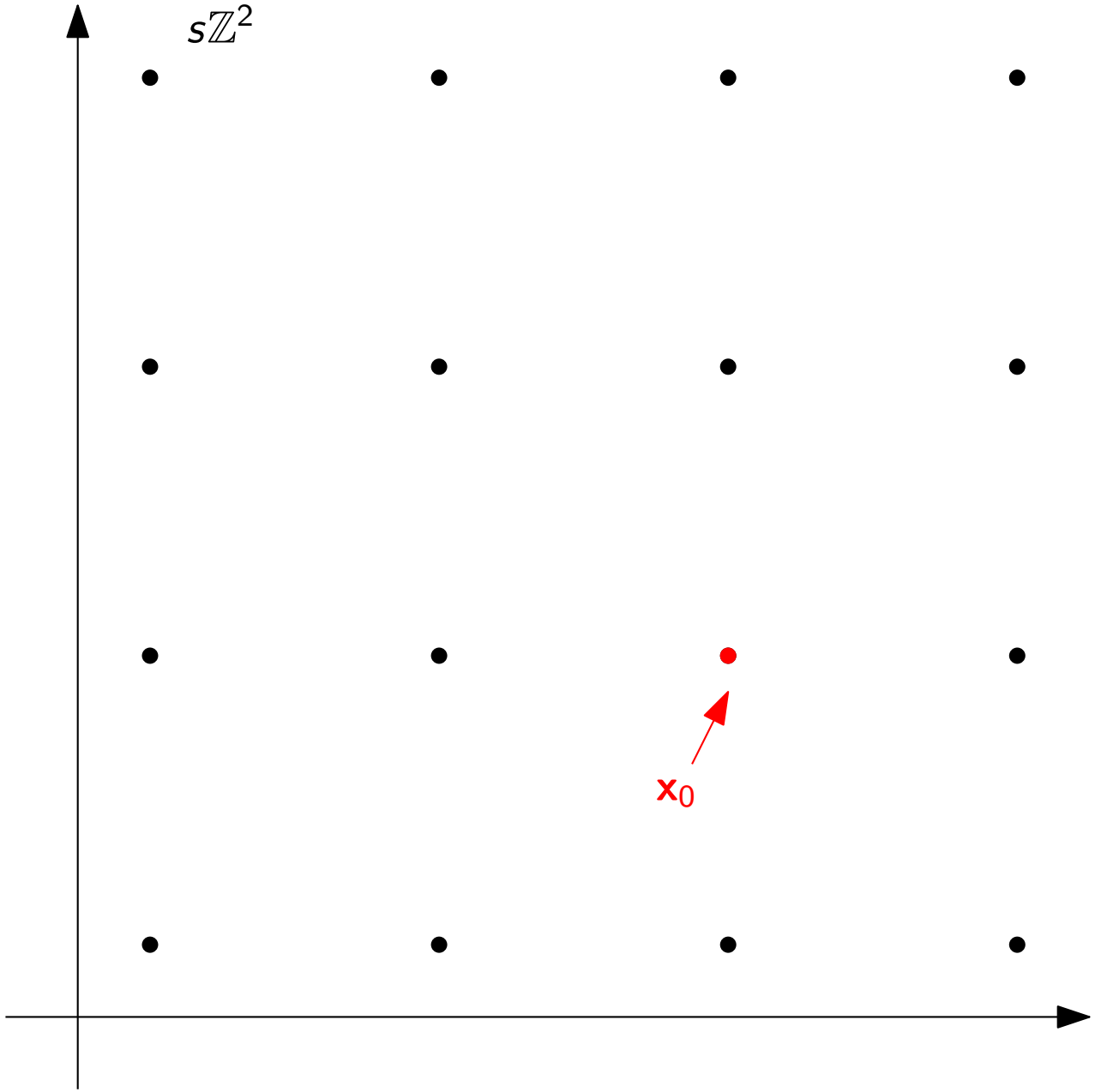
$$g(\mathrm{td}_D(A), \|A\|_\infty)\, n \log \|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty \log n .$$

- $\log \|\mathbf{b}\|_\infty \Leftarrow$ finding an initial feasible solution.
- **Conjecture:** The dependence on n in Theorem 2 is optimal.
- **Remark:** optimizing dependence on parameters $\|A\|_\infty, \mathrm{td}_P(A), \mathrm{td}_D(A)$: **NOT** focus of this work.

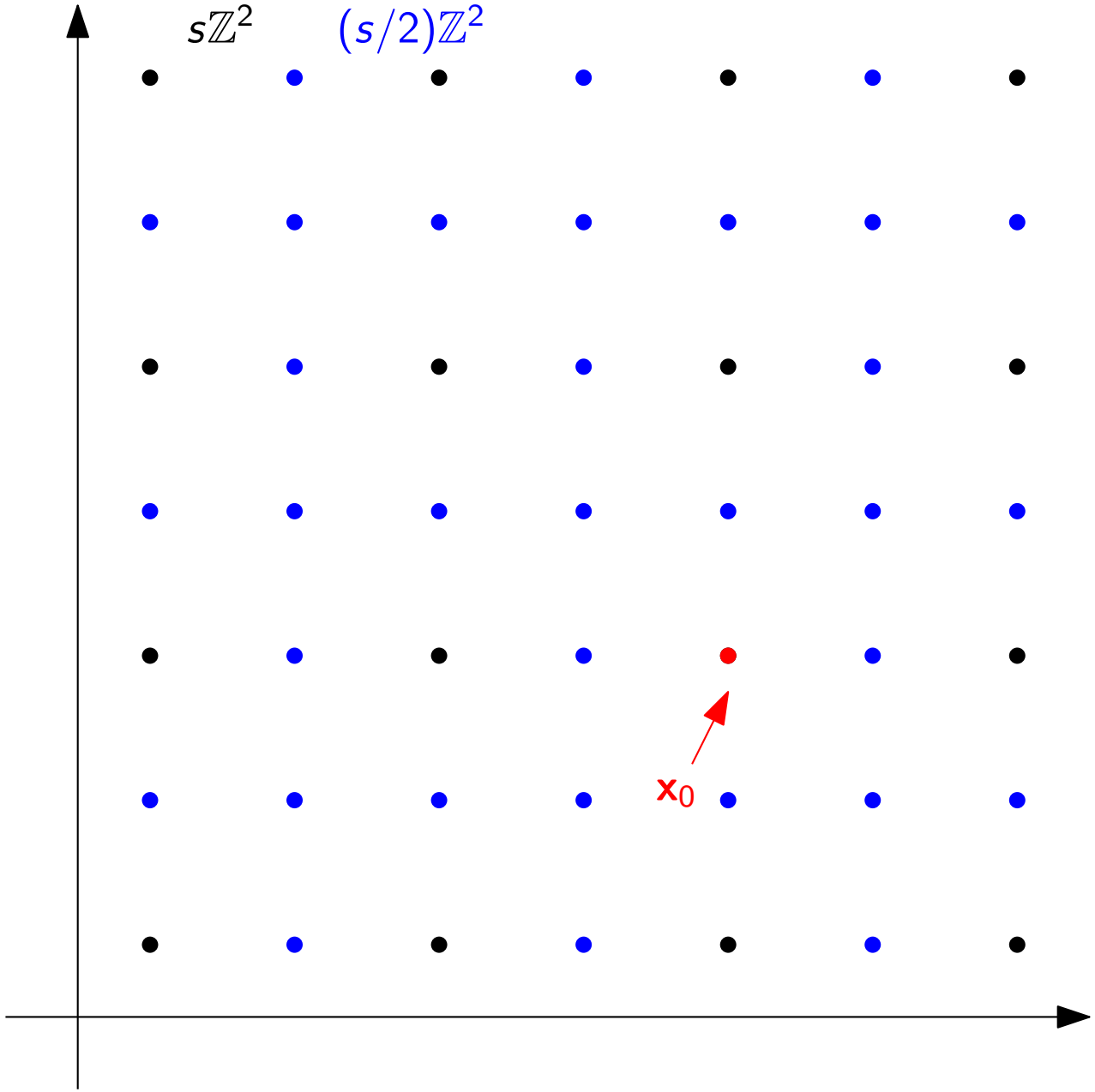
Main Idea: Scaling Algorithm



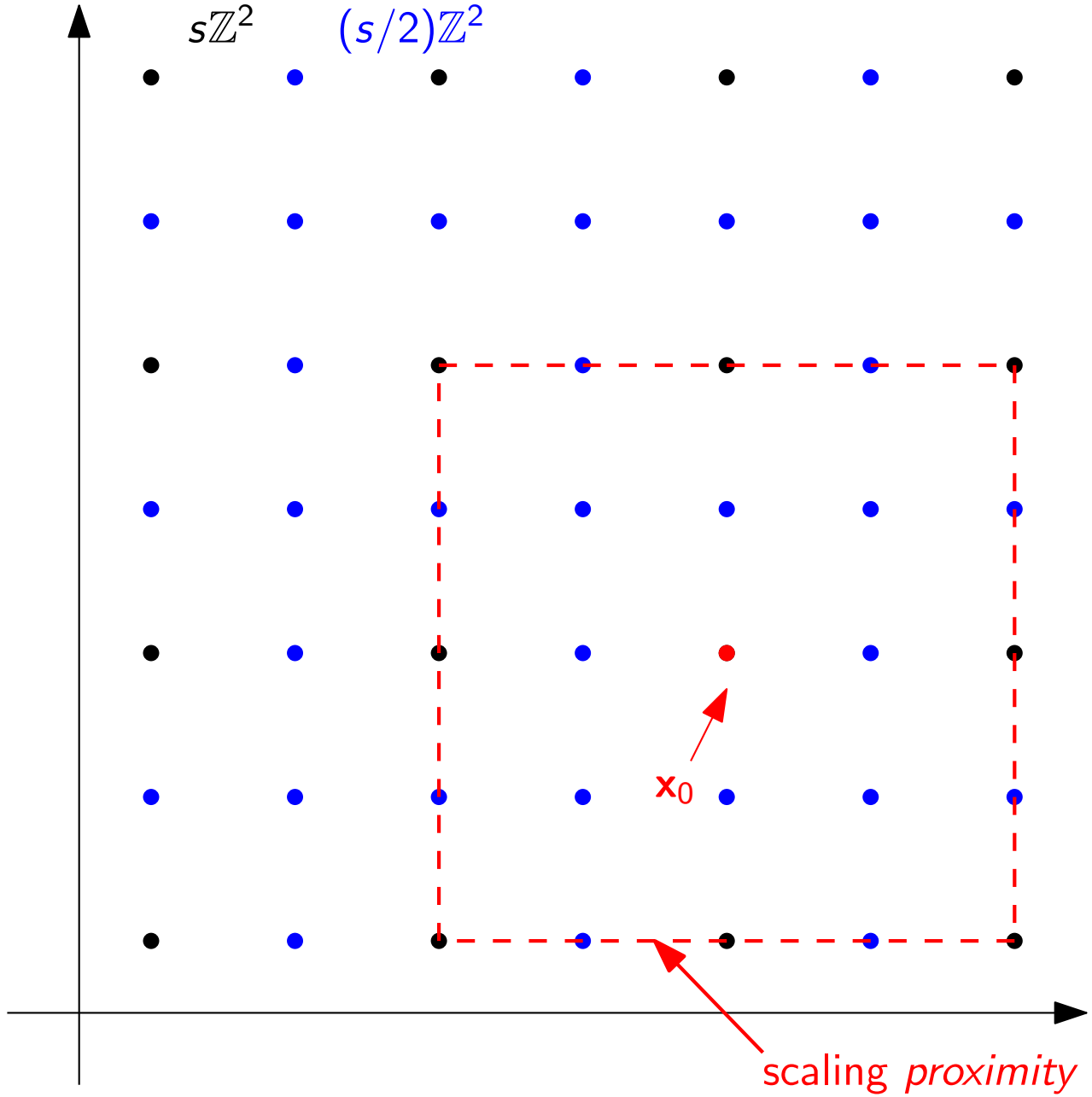
Main Idea: Scaling Algorithm



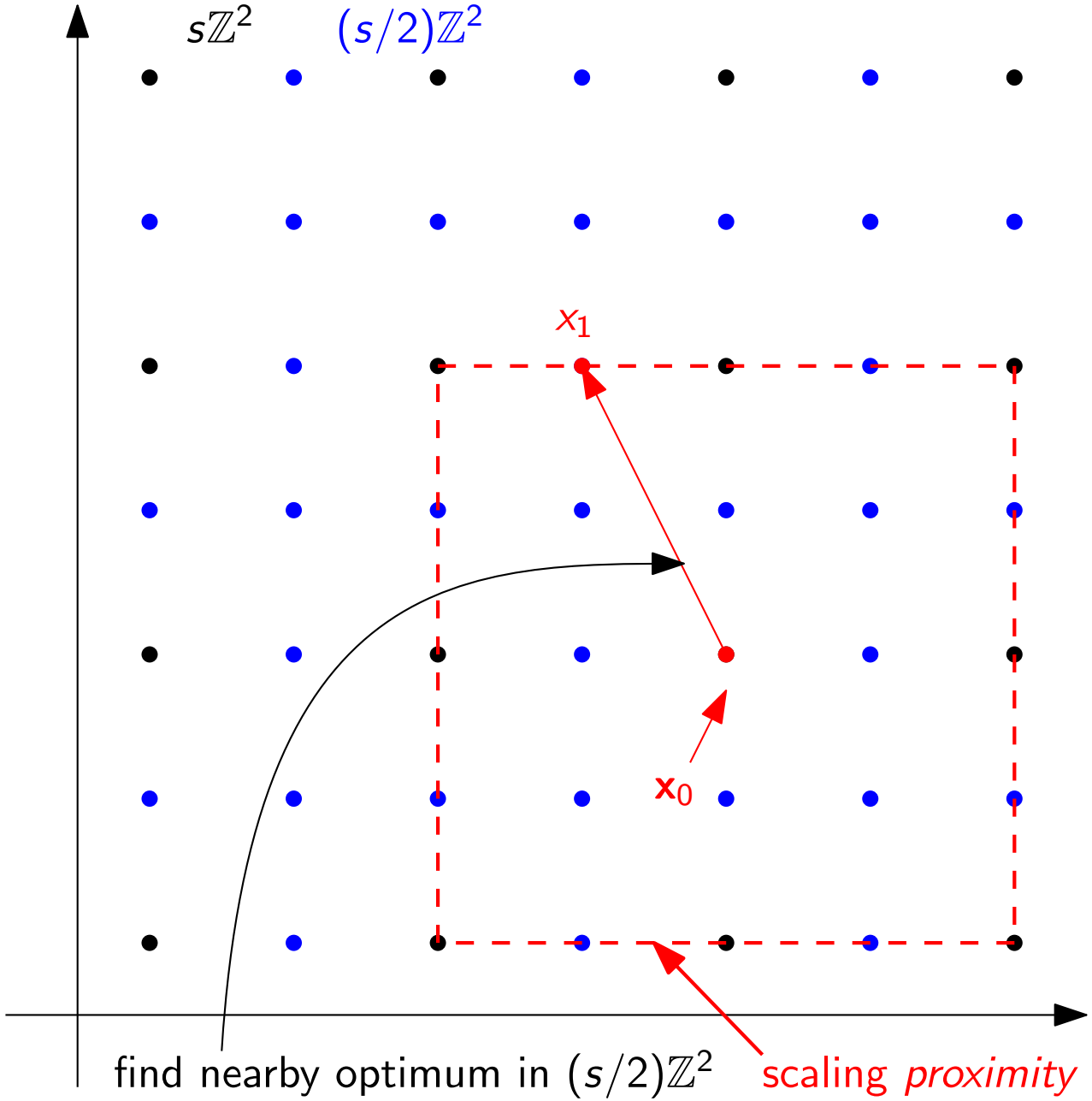
Main Idea: Scaling Algorithm



Main Idea: Scaling Algorithm



Main Idea: Scaling Algorithm



Scaling Algorithm Informally (Theorem 3)

(Given an initial feasible solution), we can solve IP by solving

Scaling Algorithm Informally (Theorem 3)

(Given an initial feasible solution), we can solve IP by solving

- $\lceil \log \|\mathbf{l}, \mathbf{u}\|_\infty \rceil + 1$ instances of IP

Scaling Algorithm Informally (Theorem 3)

(Given an initial feasible solution), we can solve IP by solving

- $\lceil \log \|\mathbf{l}, \mathbf{u}\|_\infty \rceil + 1$ instances of IP
- with small bounds $\|\mathbf{u}^i - \mathbf{l}^i\|_\infty \leq 3\rho$ where

Scaling Algorithm Informally (Theorem 3)

(Given an initial feasible solution), we can solve IP by solving

- $\lceil \log \|\mathbf{l}, \mathbf{u}\|_\infty \rceil + 1$ instances of IP
- with small bounds $\|\mathbf{u}^i - \mathbf{l}^i\|_\infty \leq 3\rho$ where
- ρ only depends on A.

Scaling Algorithm Informally (Theorem 3)

(Given an initial feasible solution), we can solve IP by solving

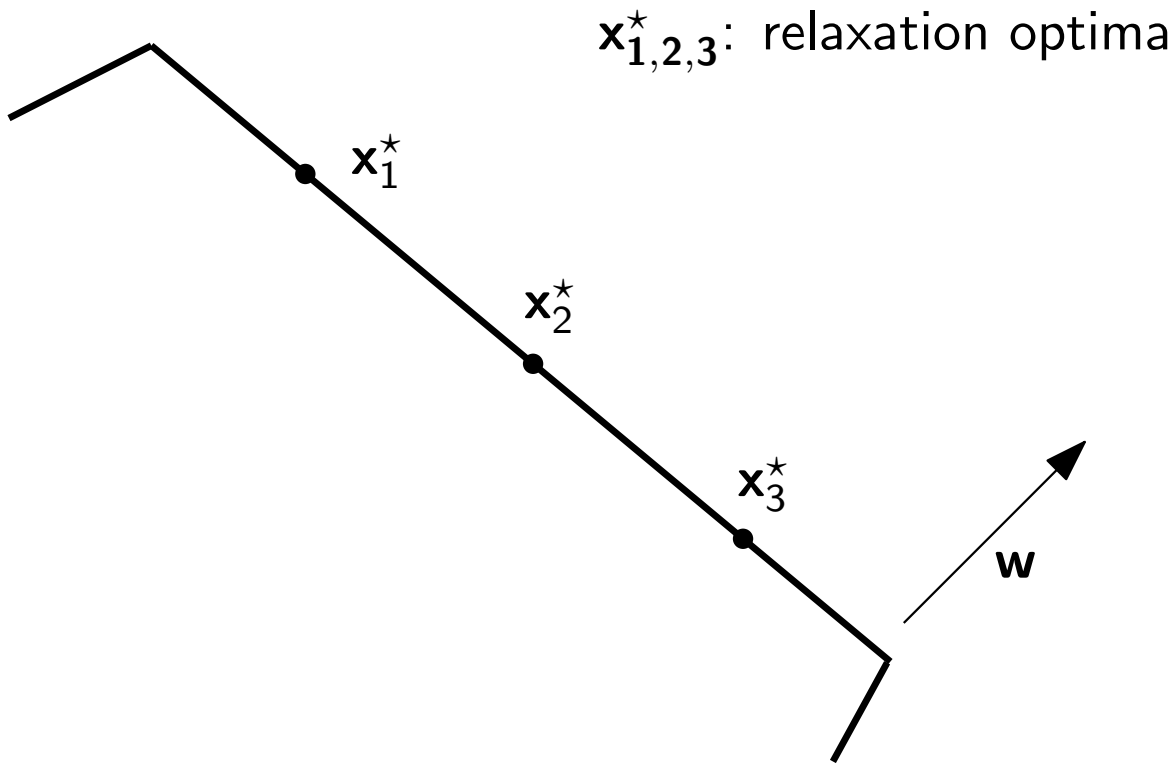
- $\lceil \log \|\mathbf{l}, \mathbf{u}\|_\infty \rceil + 1$ instances of IP
- with small bounds $\|\mathbf{u}^i - \mathbf{l}^i\|_\infty \leq 3\rho$ where
- ρ only depends on A.

Question. What is ρ?

Conformal Proximity Bound (CPB) (very informally)

$\mathcal{P}_p(A)$: given A, smallest real $r \in \mathbb{R}$ such that

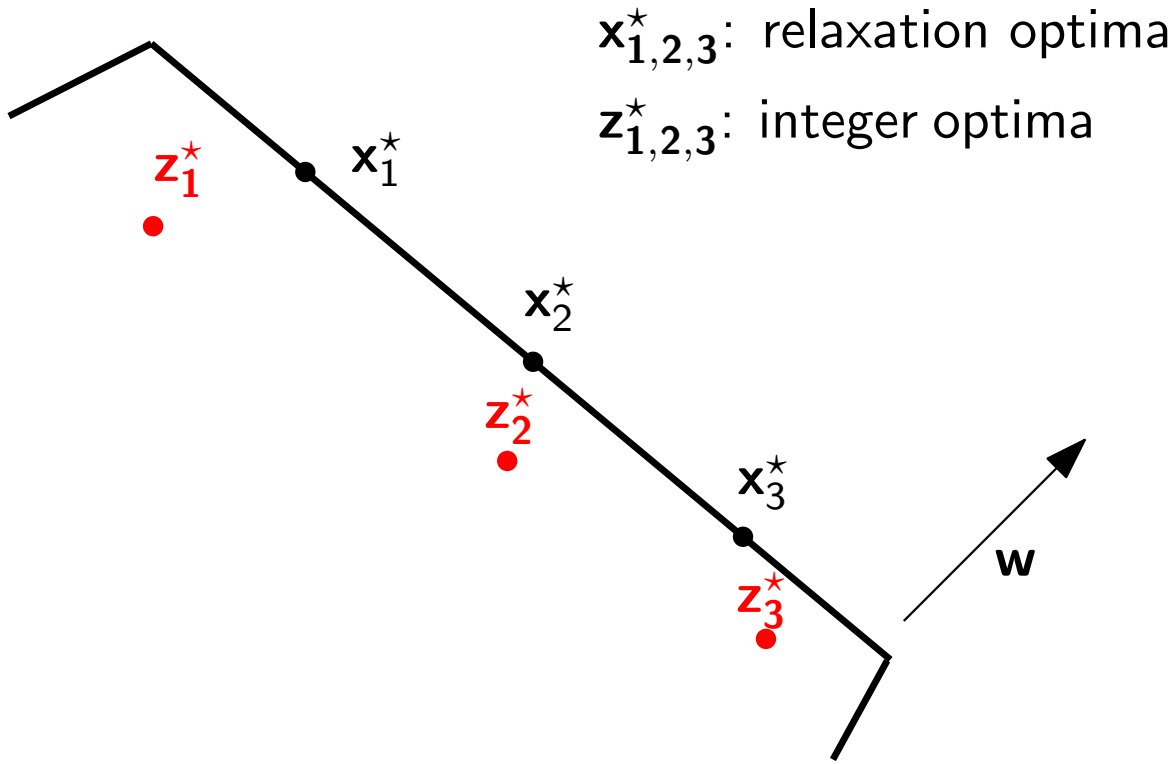
- for every relaxation optimum $\mathbf{x}^\star$



Conformal Proximity Bound (CPB) (very informally)

$\mathcal{P}_p(A)$: given A, smallest real $r \in \mathbb{R}$ such that

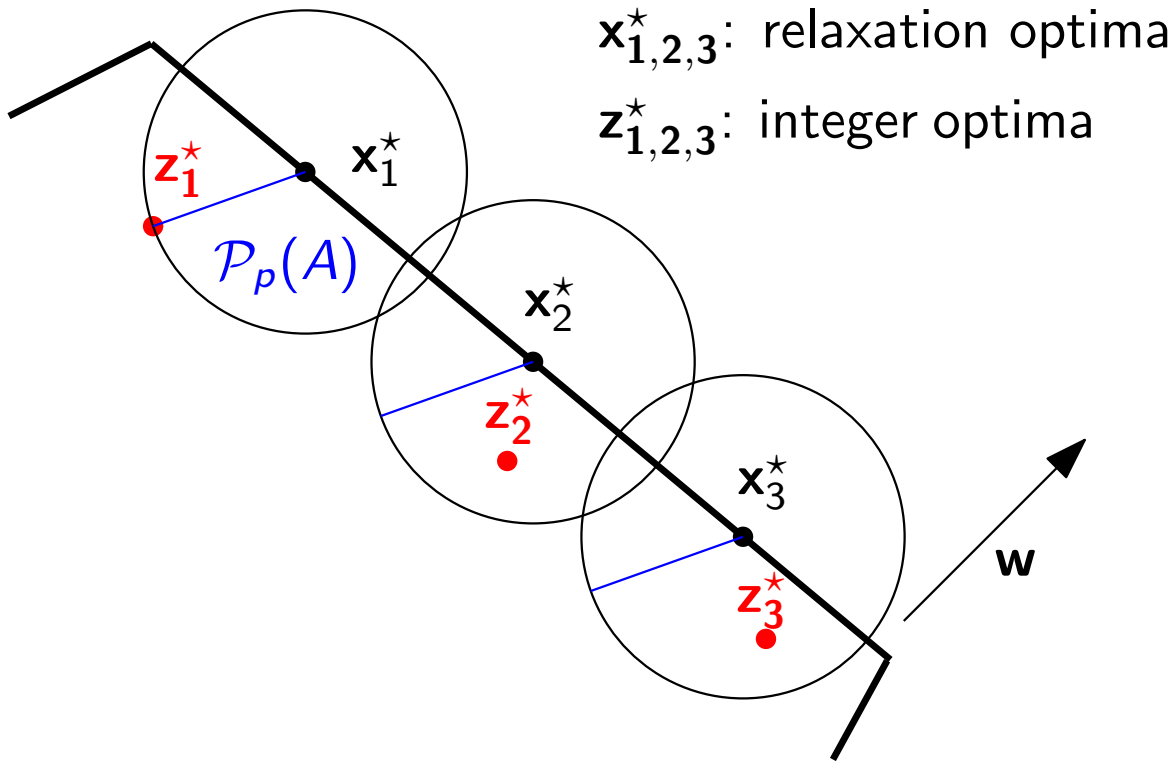
- for every relaxation optimum $\mathbf{x}^\star$
- there exists an integer optimum $\mathbf{z}^\star$



Conformal Proximity Bound (CPB) (very informally)

$\mathcal{P}_p(A)$: given A, smallest real $r \in \mathbb{R}$ such that

- for every relaxation optimum $\mathbf{x}^\star$
- there exists an integer optimum $\mathbf{z}^\star$
- with $\|\mathbf{x}^\star - \mathbf{z}^\star\|_p \leq r$.



Scaling Proximity Theorem

Theorem (Theorem 4)

- $1 \leq p \leq +\infty$
- $\mathcal{I}$: *IP instance with optimum* $\mathbf{z}^\star$

Scaling Proximity Theorem

Theorem (Theorem 4)

- $1 \leq p \leq +\infty$
- $\mathcal{I}$: *IP instance with optimum* $\mathbf{z}^\star$
- $\mathcal{I}'$: *copy of* $\mathcal{I}$ *but require that solutions be in* $s\mathbb{Z}^n$

Scaling Proximity Theorem

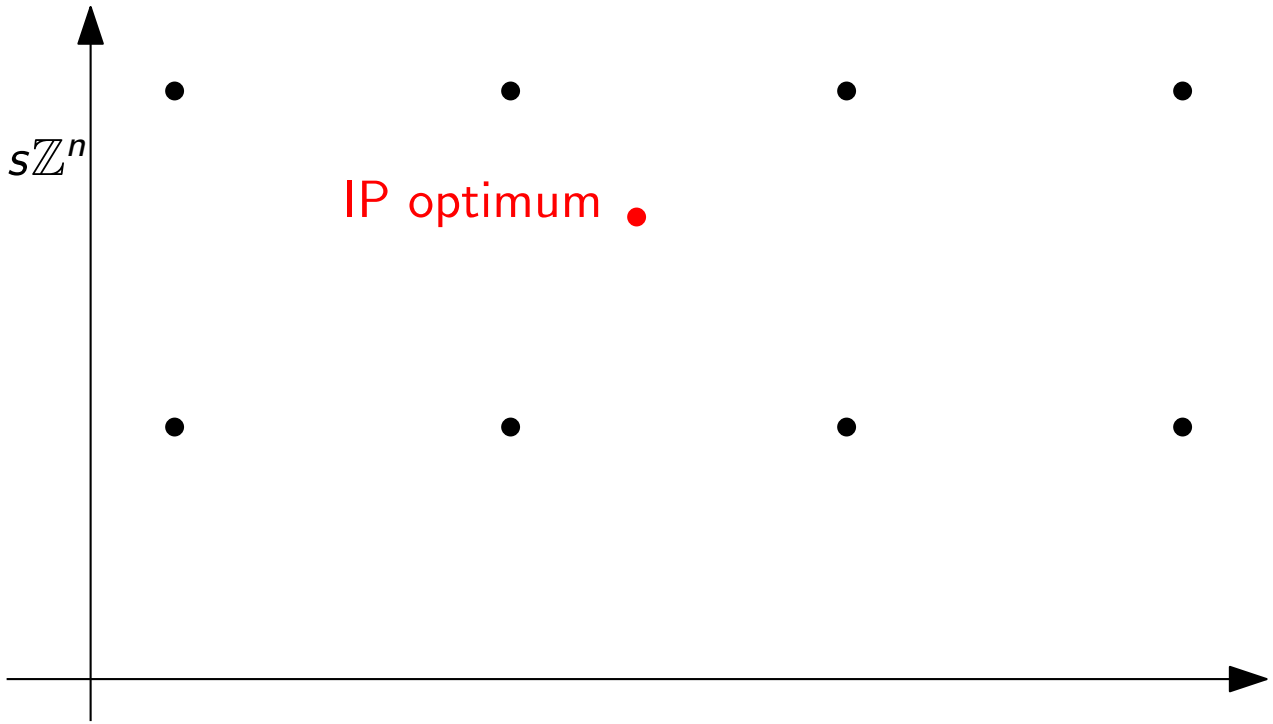
Theorem (Theorem 4)

- $1 \leq p \leq +\infty$
- $\mathcal{I}$: *IP instance with optimum* $\mathbf{z}^\star$
- $\mathcal{I}'$: *copy of* $\mathcal{I}$ *but require that solutions be in* $s\mathbb{Z}^n$
- *then there exists a solution* $\mathbf{z}'$ *of* $\mathcal{I}'$ *with*

$$\|\mathbf{z}^\star - \hat{\mathbf{z}}\|_p \leq (s+1)\mathcal{P}_p(A),$$

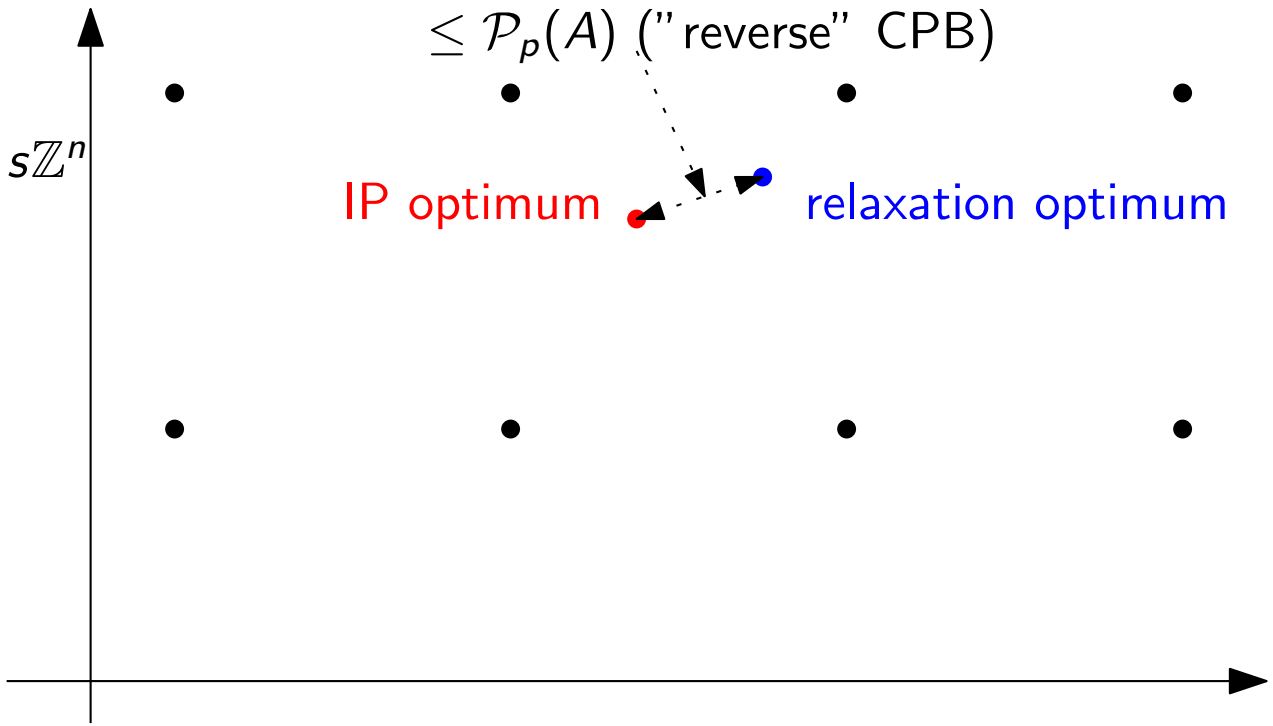
Scaling Proximity Theorem: Proof Idea

Goal: bound distance between IP optimum and optimum restricted to $s\mathbb{Z}^n$



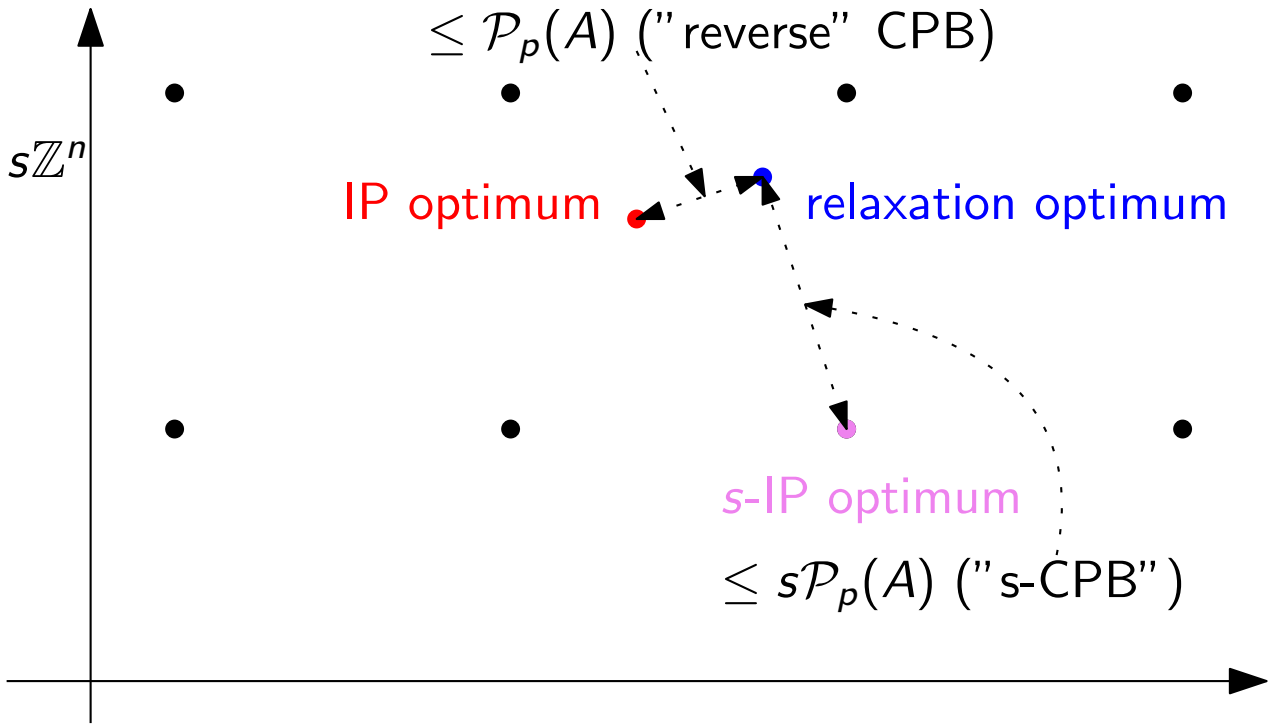
Scaling Proximity Theorem: Proof Idea

Goal: bound distance between IP optimum and optimum restricted to $s\mathbb{Z}^n$



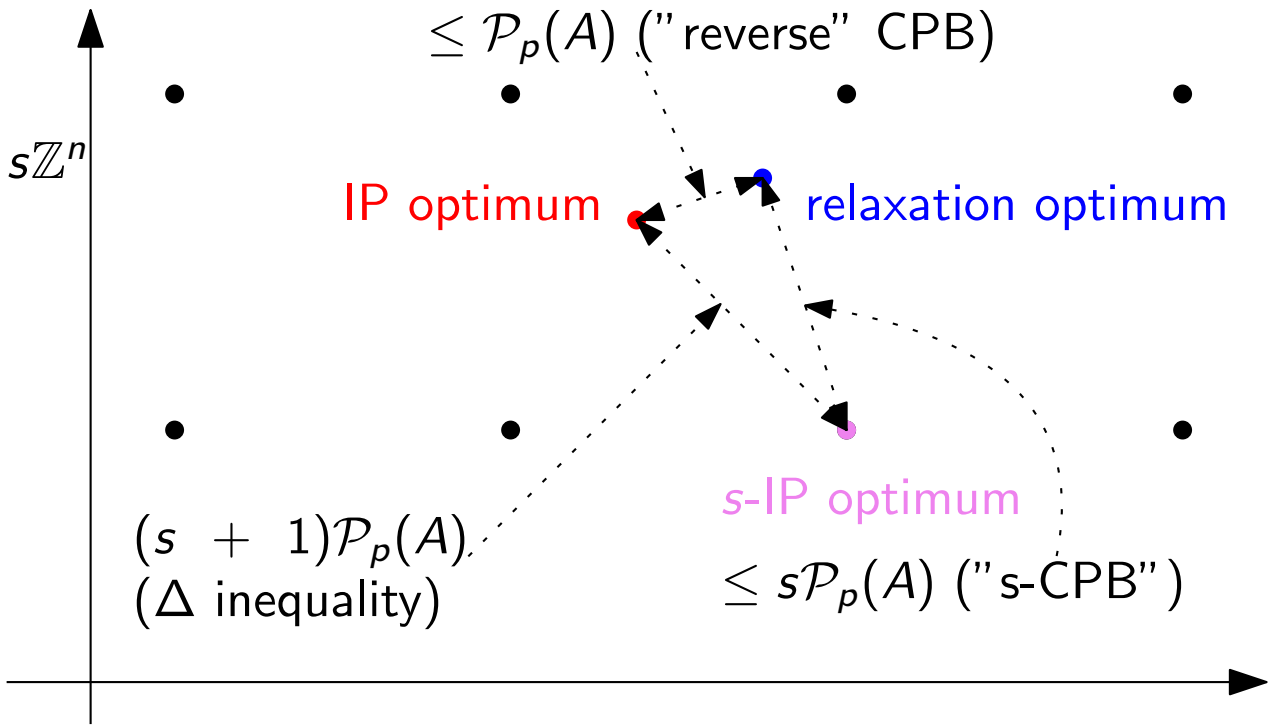
Scaling Proximity Theorem: Proof Idea

Goal: bound distance between IP optimum and optimum restricted to $s\mathbb{Z}^n$



Scaling Proximity Theorem: Proof Idea

Goal: bound distance between IP optimum and optimum restricted to $s\mathbb{Z}^n$



Scaling Algorithm: Practical takeaway

Suppose you have some class of IP's.

Scaling Algorithm: Practical takeaway

Suppose you have some class of IP's.

1. Find an upper bound ρ on $\mathcal{P}_\infty(A)$.

Scaling Algorithm: Practical takeaway

Suppose you have some class of IP's.

1. Find an upper bound ρ on $\mathcal{P}_\infty(A)$.
2. Find a way to solve IP with small bounds.

Scaling Algorithm: Practical takeaway

Suppose you have some class of IP's.

1. Find an upper bound ρ on $\mathcal{P}_\infty(A)$.
2. Find a way to solve IP with small bounds.
3. Plug these into scaling algorithm.

Scaling Algorithm: Practical takeaway

Suppose you have some class of IP's.

1. Find an upper bound ρ on $\mathcal{P}_\infty(A)$.
2. Find a way to solve IP with small bounds.
3. Plug these into scaling algorithm.
4. Profit (=get to solve $\lceil \log \|\mathbf{l}, \mathbf{u}\|_\infty \rceil + 1$ simple instances instead of one complicated one).

Scaling Algorithm: Practical takeaway

Suppose you have some class of IP's.

1. Find an upper bound ρ on $\mathcal{P}_\infty(A)$.
2. Find a way to solve IP with small bounds.
3. Plug these into scaling algorithm.
4. Profit (=get to solve $\lceil \log \|\mathbf{l}, \mathbf{u}\|_\infty \rceil + 1$ simple instances instead of one complicated one).

Remark: Number of IP instances can be reduced to $\log \lceil \|\mathbf{l}, \mathbf{u}\|_\infty / \rho \rceil + 1$.

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty)$-time algorithm)

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u} - \mathbf{l}, \mathbf{b}\|_\infty)$-time algorithm)

[Klein, Reuter; SODA 2022]

There is a computable function g' such that

$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty)$$

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm)

[Klein, Reuter; SODA 2022]

There is a computable function g' such that

$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty)$$

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024, Lemma 8]

Each IP instance in the scaling algorithm can be solved in time

$$\mathcal{O}(\mathrm{td}_P(A)^2(2\mathcal{P}_p(A)+1)^{\mathrm{td}_P(A)}n)$$

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm)

[Klein, Reuter; SODA 2022]

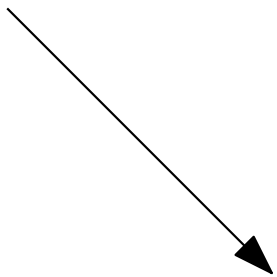
There is a computable function g' such that

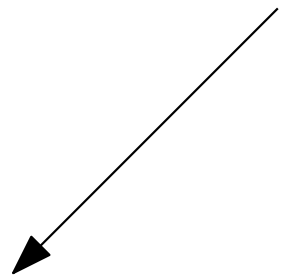
$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty)$$

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024, Lemma 8]

Each IP instance in the scaling algorithm can be solved in time

$$\mathcal{O}(\mathrm{td}_P(A)^2(2\mathcal{P}_p(A)+1)^{\mathrm{td}_P(A)}n)$$





$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n)$-time algorithm IP with small bounds

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm)

[Klein, Reuter; SODA 2022]

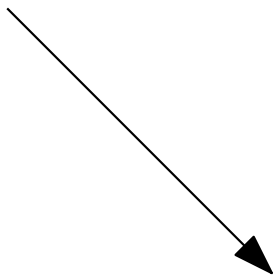
There is a computable function g' such that

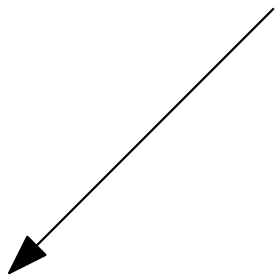
$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty)$$

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024, Lemma 8]

Each IP instance in the scaling algorithm can be solved in time

$$\mathcal{O}(\mathrm{td}_P(A)^2(2\mathcal{P}_p(A)+1)^{\mathrm{td}_P(A)}n)$$





$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n)$-time algorithm IP with small bounds

$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n \log \|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm for IP

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm)

[Klein, Reuter; SODA 2022]

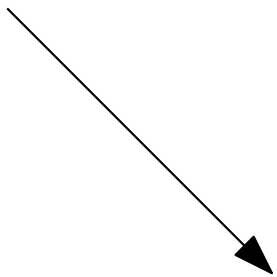
There is a computable function g' such that

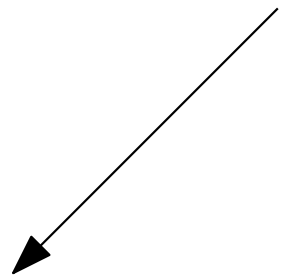
$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty)$$

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024, Lemma 8]

Each IP instance in the scaling algorithm can be solved in time

$$\mathcal{O}(\mathrm{td}_P(A)^2(2\mathcal{P}_p(A)+1)^{\mathrm{td}_P(A)}n)$$





$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n)$-time algorithm IP with small bounds

$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n \log \|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm for IP

- g' is triple exponential in $\mathrm{td}_P(A)$.

Primal Algorithm (Theorem 1) ($\mathcal{O}(g(\|A\|_\infty, \mathrm{td}_P(A))n\|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm)

[Klein, Reuter; SODA 2022]

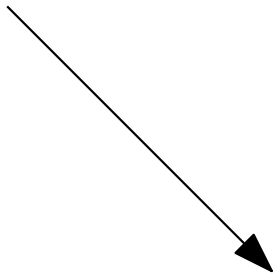
There is a computable function g' such that

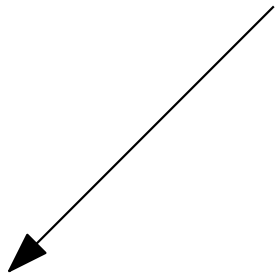
$$\mathcal{P}_\infty(A) \leq g'(\mathrm{td}_P(A), \|A\|_\infty)$$

[Eisenbrand, Hunkenschröder, Klein, Koutecký, Levin, Onn; MOR 2024, Lemma 8]

Each IP instance in the scaling algorithm can be solved in time

$$\mathcal{O}(\mathrm{td}_P(A)^2(2\mathcal{P}_p(A)+1)^{\mathrm{td}_P(A)}n)$$





$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n)$-time algorithm IP with small bounds

$\mathcal{O}(g''(\mathrm{td}_P(A), \|A\|_\infty)n \log \|\mathbf{u}-\mathbf{l},\mathbf{b}\|_\infty)$-time algorithm for IP

- g' is triple exponential in $\mathrm{td}_P(A)$.
  - Seems optimal [Hunkenschröder, Klein, Koutecký, Lassota, Levin; IPCO 2024, Theorem 1]

Dual Algorithm (Theorem 2)

More delicate than Theorem 1:

Dual Algorithm (Theorem 2)

More delicate than Theorem 1:

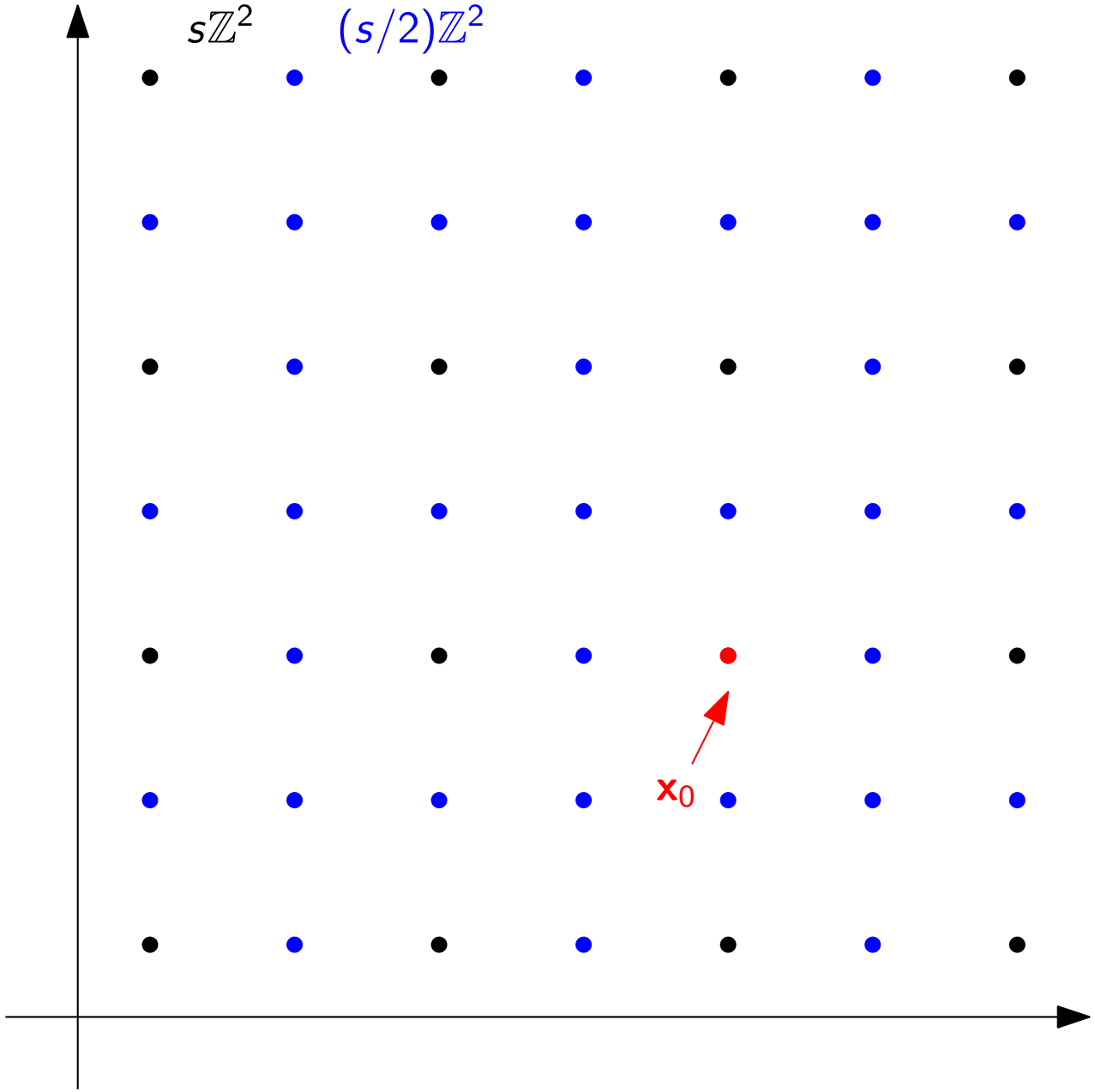
- No analogous result to the proximity result of Klein and Reuter ($\mathcal{P}_\infty(A) \leq g'(\mathrm{td}(A), \|A\|_\infty)$).

Dual Algorithm (Theorem 2)

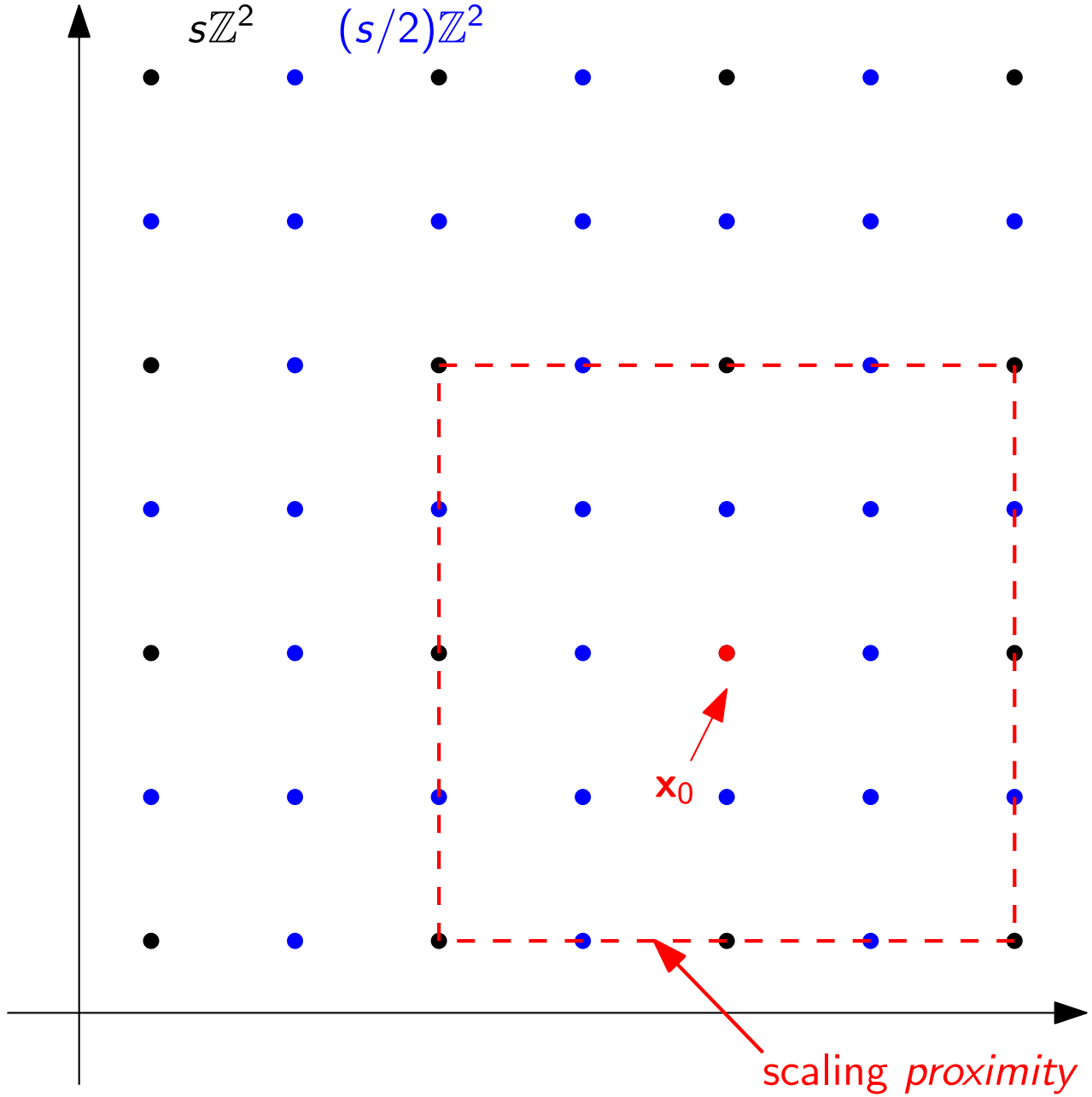
More delicate than Theorem 1:

- No analogous result to the proximity result of Klein and Reuter ($\mathcal{P}_\infty(A) \leq g'(\mathrm{td}(A), \|A\|_\infty)$).
- [Cslovjecsek, Eisenbrand, Hunkenschröder, Rohwedder, Weismantel; SODA 2021, Proposition 4.1] shows instances of IP with
  - small $\mathrm{td}_D(A) + \|A\|_\infty$
  - but where integer optima are $\Omega(n)$ far in the ℓ_∞-norm from any continuous optimum.

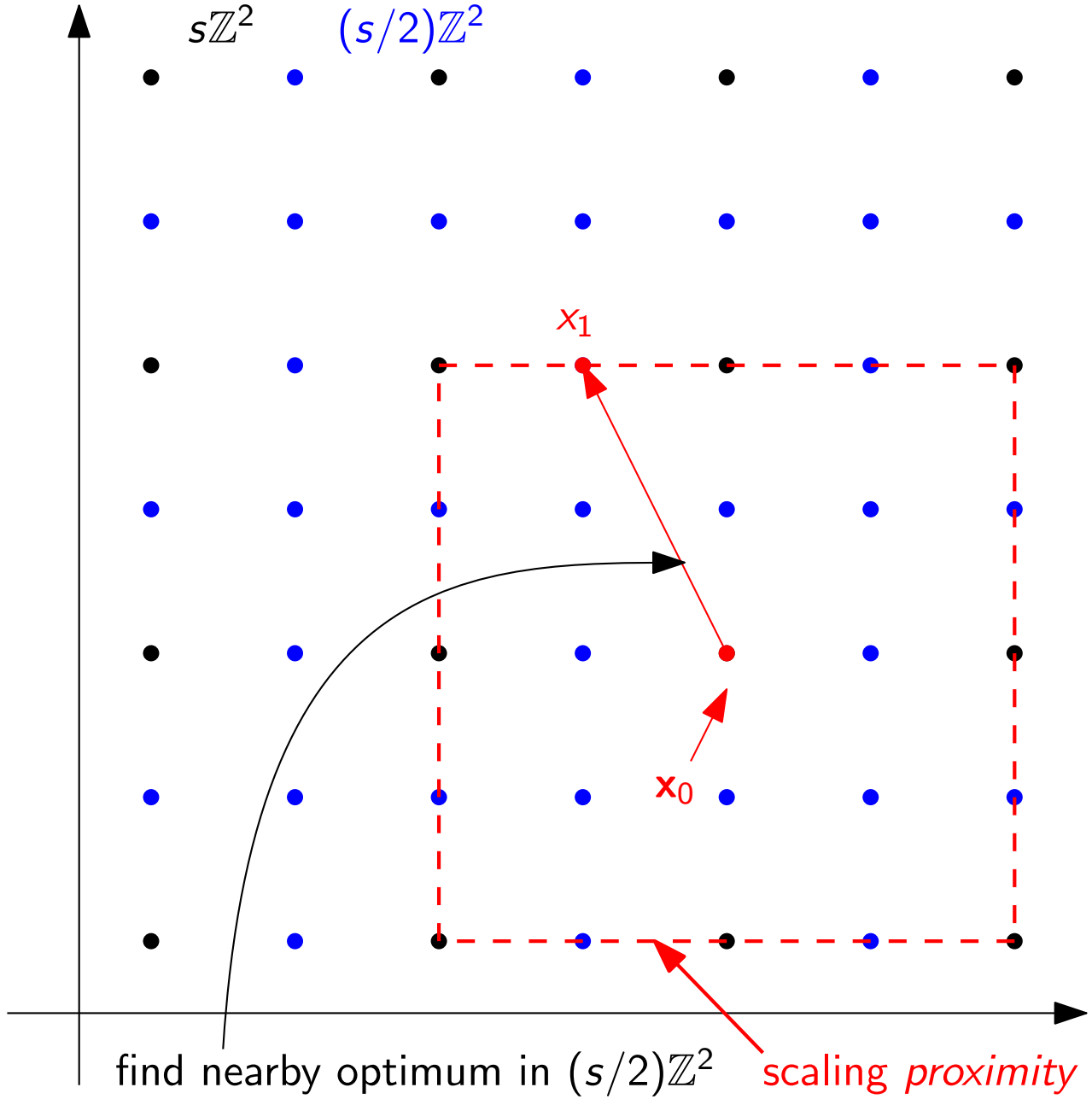
Dual Algorithm: Reexamine the Scaling



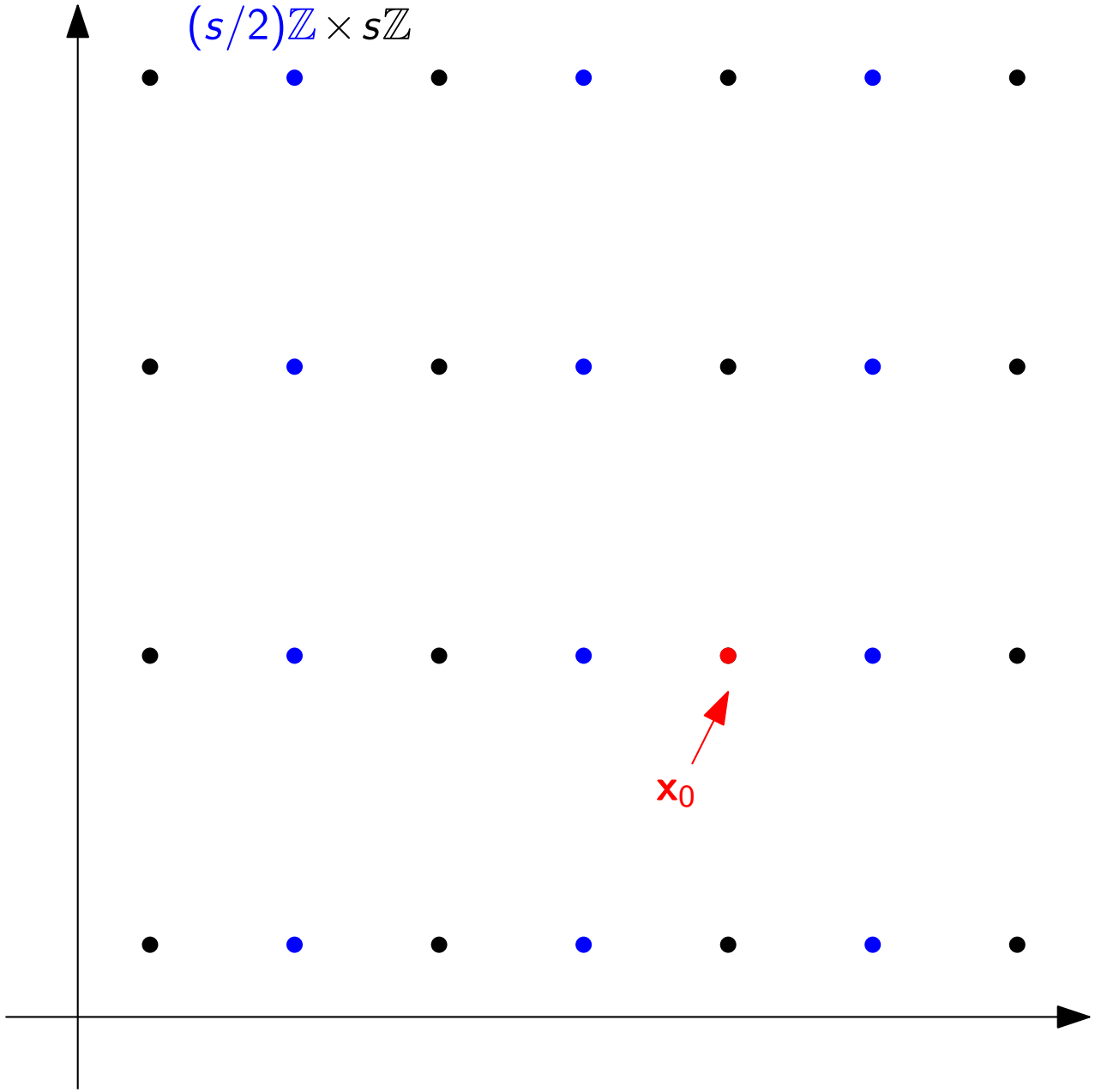
Dual Algorithm: Reexamine the Scaling



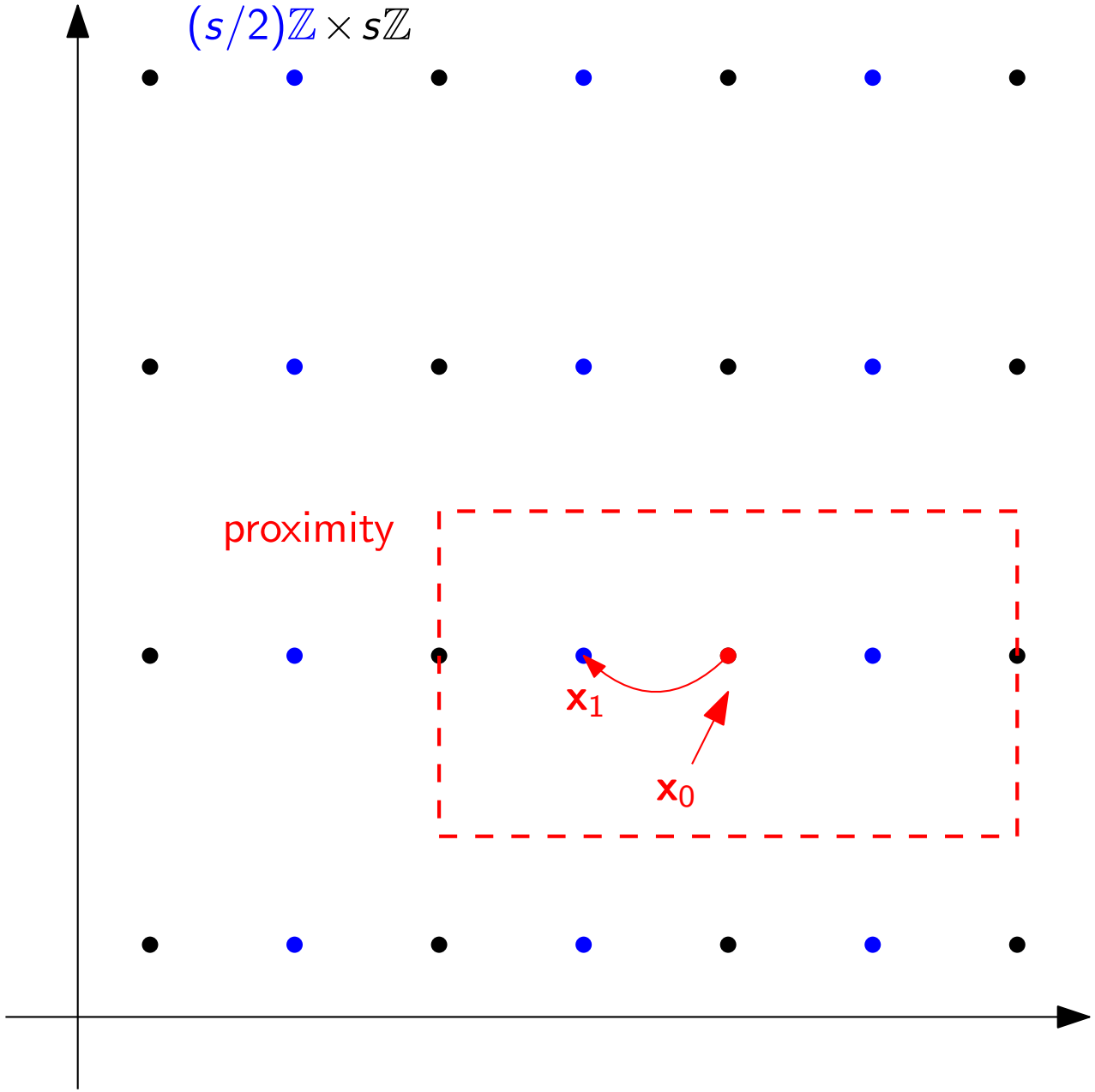
Dual Algorithm: Reexamine the Scaling



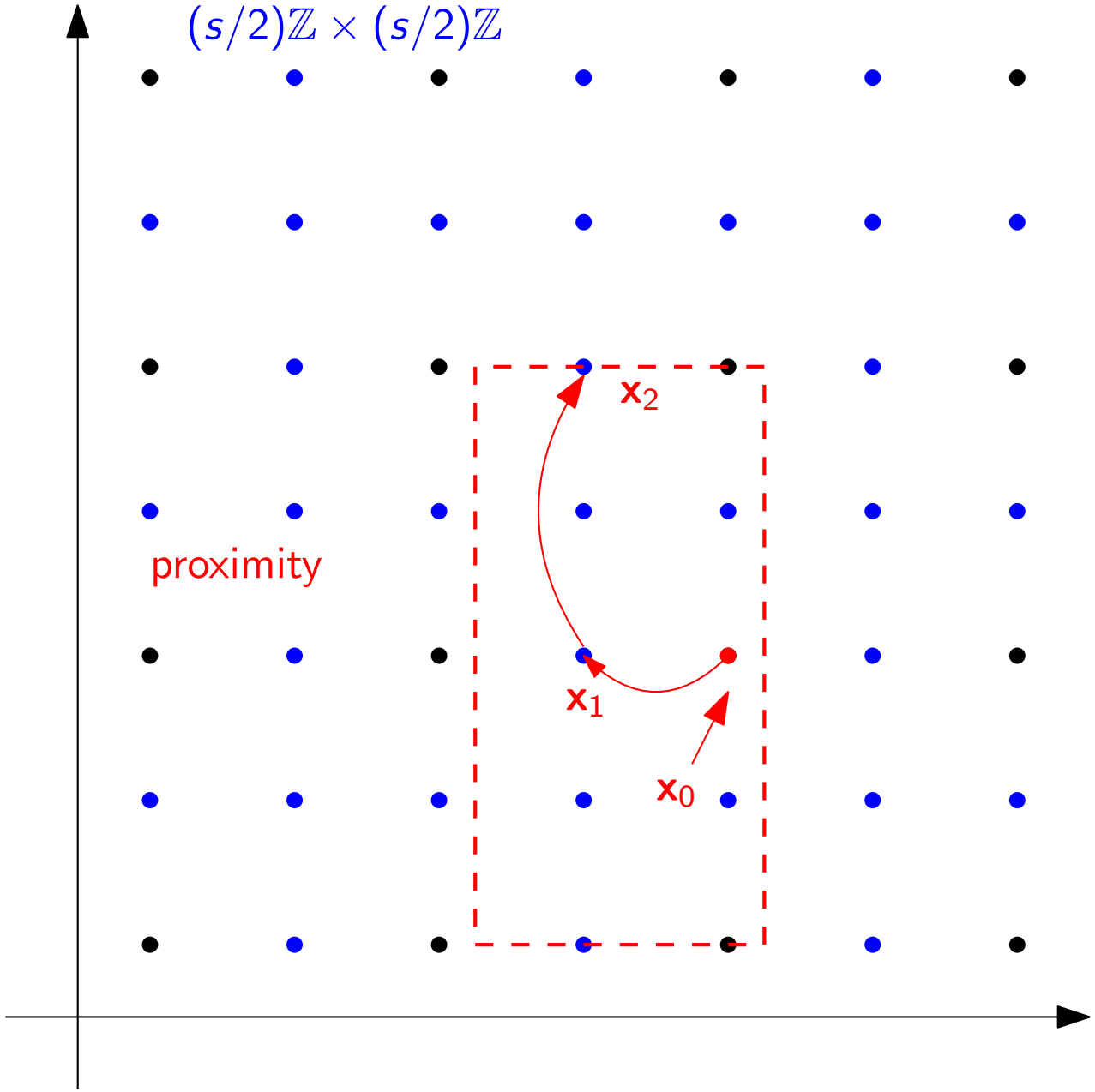
Dual Algorithm: Reexamine the Scaling



Dual Algorithm: Reexamine the Scaling



Dual Algorithm: Reexamine the Scaling



Dual Algorithm: Proximity of densifying one by one

Corollary (Informal corollary of Theorem 6)

- $\mathbf{x}^\star$: *optimum of IP with some (or none) variables densified*

Dual Algorithm: Proximity of densifying one by one

Corollary (Informal corollary of Theorem 6)

- $\mathbf{x}^\star$: *optimum of IP with some (or none) variables densified*
- *if we densify another variable*

Dual Algorithm: Proximity of densifying one by one

Corollary (Informal corollary of Theorem 6)

- $\mathbf{x}^\star$: *optimum of IP with some (or none) variables densified*
- *if we densify another variable*
- *then the new instance has an optimum* $\hat{\mathbf{x}}$

Dual Algorithm: Proximity of densifying one by one

Corollary (Informal corollary of Theorem 6)

- $\mathbf{x}^\star$: *optimum of IP with some (or none) variables densified*
- *if we densify another variable*
- *then the new instance has an optimum* $\hat{\mathbf{x}}$
- *with* $\|\mathbf{x}^\star - \hat{\mathbf{x}}\|_1 \leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$.

Dual Algorithm: Using the corollary

- Do the scaling algorithm.

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one
    - Theorem 6 $\Rightarrow$ new optimum is $\leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$ far in ℓ_1-norm

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one
    - Theorem 6 $\Rightarrow$ new optimum is $\leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$ far in ℓ_1-norm

Densifying a variable $\Rightarrow$ following IP where $\mathbf{x}$ is a given initial feasible solution.

$$\min\{f(\mathbf{x}+\mathbf{h}) \mid A\mathbf{h} = \mathbf{0},\ \mathbf{l} \leq \mathbf{x}+\mathbf{h} \leq \mathbf{u},\ \|\mathbf{h}\|_1 \leq \rho,\ \mathbf{h} \in \mathbb{Z}^n\} \quad . \tag{ℓ_1-IP}$$

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one
    - Theorem 6 $\Rightarrow$ new optimum is $\leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$ far in ℓ_1-norm

Densifying a variable $\Rightarrow$ following IP where $\mathbf{x}$ is a given initial feasible solution.

$$\min\{f(\mathbf{x}+\mathbf{h}) \mid A\mathbf{h} = \mathbf{0},\ \mathbf{l} \leq \mathbf{x}+\mathbf{h} \leq \mathbf{u},\ \|\mathbf{h}\|_1 \leq \rho,\ \mathbf{h} \in \mathbb{Z}^n\} \quad . \tag{ℓ_1-IP}$$

- ℓ_1-IP can be solved in time $g(\|A\|_\infty, \mathrm{td}_D(A))n$ via DP.

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one
    - Theorem 6 $\Rightarrow$ new optimum is $\leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$ far in ℓ_1-norm

Densifying a variable $\Rightarrow$ following IP where $\mathbf{x}$ is a given initial feasible solution.

$$\min\{f(\mathbf{x}+\mathbf{h}) \mid A\mathbf{h} = \mathbf{0},\ \mathbf{l} \leq \mathbf{x}+\mathbf{h} \leq \mathbf{u},\ \|\mathbf{h}\|_1 \leq \rho,\ \mathbf{h} \in \mathbb{Z}^n\} \quad . \tag{ℓ_1-IP}$$

- ℓ_1-IP can be solved in time $g(\|A\|_\infty, \mathrm{td}_D(A))n$ via DP.
- Leads to linear time per variable $\Rightarrow$ quadratic in n overall.

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one
    - Theorem 6 $\Rightarrow$ new optimum is $\leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$ far in ℓ_1-norm

Densifying a variable $\Rightarrow$ following IP where $\mathbf{x}$ is a given initial feasible solution.

$$\min\{f(\mathbf{x}+\mathbf{h}) \mid A\mathbf{h} = \mathbf{0},\ \mathbf{l} \leq \mathbf{x}+\mathbf{h} \leq \mathbf{u},\ \|\mathbf{h}\|_1 \leq \rho,\ \mathbf{h} \in \mathbb{Z}^n\} \quad . \tag{ℓ_1-IP}$$

- ℓ_1-IP can be solved in time $g(\|A\|_\infty, \mathrm{td}_D(A))n$ via DP.
- Leads to linear time per variable $\Rightarrow$ quadratic in n overall.
- Instead, we "dynamize" the DP table so that densifying a single variable can be done in time $g'(\|A\|_\infty, \mathrm{td}_D(A)) \log n$.

Dual Algorithm: Using the corollary

- Do the scaling algorithm.
- In each iteration
  - × densify ~~all variables~~
  - ✓ densify variables one by one
    - Theorem 6 $\Rightarrow$ new optimum is $\leq 4g(\|A\|_\infty, \mathrm{td}_D(A))$ far in ℓ_1-norm

Densifying a variable $\Rightarrow$ following IP where $\mathbf{x}$ is a given initial feasible solution.

$$\min\{f(\mathbf{x}+\mathbf{h}) \mid A\mathbf{h} = \mathbf{0},\ \mathbf{l} \leq \mathbf{x}+\mathbf{h} \leq \mathbf{u},\ \|\mathbf{h}\|_1 \leq \rho,\ \mathbf{h} \in \mathbb{Z}^n\} \quad . \tag{ℓ_1-IP}$$

- ℓ_1-IP can be solved in time $g(\|A\|_\infty, \mathrm{td}_D(A))n$ via DP.
- Leads to linear time per variable $\Rightarrow$ quadratic in n overall.
- Instead, we "dynamize" the DP table so that densifying a single variable can be done in time $g'(\|A\|_\infty, \mathrm{td}_D(A)) \log n$.
- Overall dependence on n is $n \log n$.

Concluding Remarks & Open Problems

Remarks for Dual Algorithm

- Our approach for the dual algorithm cannot be sped up (by reduction from sorting in the comparison model).

Concluding Remarks & Open Problems

Remarks for Dual Algorithm

- Our approach for the dual algorithm cannot be sped up (by reduction from sorting in the comparison model).
- **Conjecture.** The dependence on n in the running time of the dual algorithm is optimal.

Concluding Remarks & Open Problems

Remarks for Dual Algorithm

- Our approach for the dual algorithm cannot be sped up (by reduction from sorting in the comparison model).
- **Conjecture.** The dependence on n in the running time of the dual algorithm is optimal.

Open problems

- Remove $\log n$ factor in the dual algorithm.

Concluding Remarks & Open Problems

Remarks for Dual Algorithm

- Our approach for the dual algorithm cannot be sped up (by reduction from sorting in the comparison model).
- **Conjecture.** The dependence on n in the running time of the dual algorithm is optimal.

Open problems

- Remove $\log n$ factor in the dual algorithm.
- Add $\log n$ factor to $n \log \|\mathbf{u} - \mathbf{l}\|_\infty$ lower bound.

Concluding Remarks & Open Problems

Remarks for Dual Algorithm

- Our approach for the dual algorithm cannot be sped up (by reduction from sorting in the comparison model).
- **Conjecture.** The dependence on n in the running time of the dual algorithm is optimal.

Open problems

- Remove $\log n$ factor in the dual algorithm.
- Add $\log n$ factor to $n \log \|\mathbf{u} - \mathbf{l}\|_\infty$ lower bound.
- Remove $\log n$ factor at least for some special cases.
  E.g. special objective functions such as separable quadratic.

Thank you for your attention!

https://arxiv.org/abs/2505.22212

<tung@iuuk.mff.cuni.cz>

Open problems

- Remove $\log n$ factor in the dual algorithm.
- Add $\log n$ factor to $n \log \|\mathbf{u} - \mathbf{l}\|_\infty$ lower bound.
- Remove $\log n$ factor at least for some special cases. E.g. special objective functions such as separable quadratic.