

## 8 Seidels LP-algorithm

- ▶ Suppose we want to solve  $\min\{c^T x \mid Ax \geq b; x \geq 0\}$ , where  $x \in \mathbb{R}^d$  and we have  $m$  constraints.
- ▶ In the worst-case Simplex runs in time roughly  $\mathcal{O}(m(m+d)\binom{m+d}{m}) \approx (m+d)^m$ . (slightly better bounds on the running time exist, but will not be discussed here).
- ▶ If  $d$  is much smaller than  $m$  one can do a lot better.
- ▶ In the following we develop an algorithm with running time  $\mathcal{O}(d! \cdot m)$ , i.e., linear in  $m$ .

## 8 Seidels LP-algorithm

- ▶ Suppose we want to solve  $\min\{c^T x \mid Ax \geq b; x \geq 0\}$ , where  $x \in \mathbb{R}^d$  and we have  $m$  constraints.
- ▶ In the worst-case Simplex runs in time roughly  $\mathcal{O}(m(m+d)\binom{m+d}{m}) \approx (m+d)^m$ . (slightly better bounds on the running time exist, but will not be discussed here).
- ▶ If  $d$  is much smaller than  $m$  one can do a lot better.
- ▶ In the following we develop an algorithm with running time  $\mathcal{O}(d! \cdot m)$ , i.e., linear in  $m$ .

## 8 Seidels LP-algorithm

- ▶ Suppose we want to solve  $\min\{c^T x \mid Ax \geq b; x \geq 0\}$ , where  $x \in \mathbb{R}^d$  and we have  $m$  constraints.
- ▶ In the worst-case Simplex runs in time roughly  $\mathcal{O}(m(m+d)\binom{m+d}{m}) \approx (m+d)^m$ . (slightly better bounds on the running time exist, but will not be discussed here).
- ▶ If  $d$  is much smaller than  $m$  one can do a lot better.
- ▶ In the following we develop an algorithm with running time  $\mathcal{O}(d! \cdot m)$ , i.e., linear in  $m$ .

## 8 Seidels LP-algorithm

- ▶ Suppose we want to solve  $\min\{c^T x \mid Ax \geq b; x \geq 0\}$ , where  $x \in \mathbb{R}^d$  and we have  $m$  constraints.
- ▶ In the worst-case Simplex runs in time roughly  $\mathcal{O}(m(m+d)\binom{m+d}{m}) \approx (m+d)^m$ . (slightly better bounds on the running time exist, but will not be discussed here).
- ▶ If  $d$  is much smaller than  $m$  one can do a lot better.
- ▶ In the following we develop an algorithm with running time  $\mathcal{O}(d! \cdot m)$ , i.e., **linear in  $m$** .

# 8 Seidels LP-algorithm

## Setting:

- ▶ We assume an LP of the form

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \end{array}$$

- ▶ We assume that the LP is **bounded**.

# Ensuring Conditions

Given a **standard minimization LP**

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \end{array}$$

how can we obtain an LP of the required form?

- ▶ **Compute a lower bound on  $c^T x$  for any basic feasible solution.**

# Computing a Lower Bound

Let  $s$  denote the smallest common multiple of all denominators of entries in  $A, b$ .

Multiply entries in  $A, b$  by  $s$  to obtain integral entries. This does not change the feasible region.

Add slack variables to  $A$ ; denote the resulting matrix with  $\tilde{A}$ .

If  $B$  is an optimal basis then  $x_B$  with  $\tilde{A}_B x_B = \tilde{b}$ , gives an optimal assignment to the basis variables (non-basic variables are 0).

## Computing a Lower Bound

Let  $s$  denote the smallest common multiple of all denominators of entries in  $A, b$ .

Multiply entries in  $A, b$  by  $s$  to obtain integral entries. **This does not change the feasible region.**

Add slack variables to  $A$ ; denote the resulting matrix with  $\tilde{A}$ .

If  $B$  is an optimal basis then  $x_B$  with  $\tilde{A}_B x_B = \tilde{b}$ , gives an optimal assignment to the basis variables (non-basic variables are 0).



## Computing a Lower Bound

Let  $s$  denote the smallest common multiple of all denominators of entries in  $A, b$ .

Multiply entries in  $A, b$  by  $s$  to obtain integral entries. **This does not change the feasible region.**

Add slack variables to  $A$ ; denote the resulting matrix with  $\tilde{A}$ .

If  $B$  is an optimal basis then  $x_B$  with  $\tilde{A}_B x_B = \tilde{b}$ , gives an optimal assignment to the basis variables (non-basic variables are 0).

## Computing a Lower Bound

Let  $s$  denote the smallest common multiple of all denominators of entries in  $A, b$ .

Multiply entries in  $A, b$  by  $s$  to obtain integral entries. **This does not change the feasible region.**

Add slack variables to  $A$ ; denote the resulting matrix with  $\bar{A}$ .

If  $B$  is an optimal basis then  $x_B$  with  $\bar{A}_B x_B = \bar{b}$ , gives an optimal assignment to the basis variables (non-basic variables are 0).

## Theorem 2 (Cramers Rule)

Let  $M$  be a matrix with  $\det(M) \neq 0$ . Then the solution to the system  $Mx = b$  is given by

$$x_i = \frac{\det(M_i)}{\det(M)},$$

where  $M_i$  is the matrix obtained from  $M$  by replacing the  $i$ -th column by the vector  $b$ .

**Proof:**

## Proof:

- ▶ Define

$$X_i = \begin{pmatrix} | & & | & | & | & & | \\ e_1 & \cdots & e_{i-1} & x & e_{i+1} & \cdots & e_n \\ | & & | & | & | & & | \end{pmatrix}$$

Note that expanding along the  $i$ -th column gives that  $\det(X_i) = x_i$ .

- ▶ Further, we have

$$MX_i = \begin{pmatrix} | & & | & | & | & & | \\ Me_1 & \cdots & Me_{i-1} & Mx & Me_{i+1} & \cdots & Me_n \\ | & & | & | & | & & | \end{pmatrix} = M_i$$

- ▶ Hence,

$$x_i = \det(X_i) = \frac{\det(M_i)}{\det(M)}$$

## Proof:

- ▶ Define

$$X_i = \begin{pmatrix} | & & | & | & | & & | \\ e_1 & \cdots & e_{i-1} & x & e_{i+1} & \cdots & e_n \\ | & & | & | & | & & | \end{pmatrix}$$

Note that expanding along the  $i$ -th column gives that  $\det(X_i) = x_i$ .

- ▶ Further, we have

$$MX_i = \begin{pmatrix} | & & | & | & | & & | \\ Me_1 & \cdots & Me_{i-1} & Mx & Me_{i+1} & \cdots & Me_n \\ | & & | & | & | & & | \end{pmatrix} = M_i$$

- ▶ Hence,

$$x_i = \det(X_i) = \frac{\det(M_i)}{\det(M)}$$

## Proof:

- ▶ Define

$$X_i = \begin{pmatrix} | & & | & | & | & & | \\ e_1 & \cdots & e_{i-1} & x & e_{i+1} & \cdots & e_n \\ | & & | & | & | & & | \end{pmatrix}$$

Note that expanding along the  $i$ -th column gives that  $\det(X_i) = x_i$ .

- ▶ Further, we have

$$MX_i = \begin{pmatrix} | & & | & | & | & & | \\ Me_1 & \cdots & Me_{i-1} & Mx & Me_{i+1} & \cdots & Me_n \\ | & & | & | & | & & | \end{pmatrix} = M_i$$

- ▶ Hence,

$$x_i = \det(X_i) = \frac{\det(M_i)}{\det(M)}$$

## Proof:

- ▶ Define

$$X_i = \begin{pmatrix} | & & | & | & | & & | \\ e_1 & \cdots & e_{i-1} & x & e_{i+1} & \cdots & e_n \\ | & & | & | & | & & | \end{pmatrix}$$

Note that expanding along the  $i$ -th column gives that  $\det(X_i) = x_i$ .

- ▶ Further, we have

$$MX_i = \begin{pmatrix} | & & | & | & | & & | \\ Me_1 & \cdots & Me_{i-1} & Mx & Me_{i+1} & \cdots & Me_n \\ | & & | & | & | & & | \end{pmatrix} = M_i$$

- ▶ Hence,

$$x_i = \det(X_i) = \frac{\det(M_i)}{\det(M)}$$



## Bounding the Determinant

Let  $Z$  be the maximum absolute entry occurring in  $\bar{A}$ ,  $\bar{b}$  or  $c$ . Let  $C$  denote the matrix obtained from  $\bar{A}_B$  by replacing the  $j$ -th column with vector  $\bar{b}$  (for some  $j$ ).

Observe that

$$|\det(C)|$$

## Bounding the Determinant

Let  $Z$  be the maximum absolute entry occurring in  $\bar{A}$ ,  $\bar{b}$  or  $c$ . Let  $C$  denote the matrix obtained from  $\bar{A}_B$  by replacing the  $j$ -th column with vector  $\bar{b}$  (for some  $j$ ).

Observe that

$$|\det(C)| = \left| \sum_{\pi \in S_m} \operatorname{sgn}(\pi) \prod_{1 \leq i \leq m} C_{i\pi(i)} \right|$$

# Bounding the Determinant

Let  $Z$  be the maximum absolute entry occurring in  $\bar{A}$ ,  $\bar{b}$  or  $c$ . Let  $C$  denote the matrix obtained from  $\bar{A}_B$  by replacing the  $j$ -th column with vector  $\bar{b}$  (for some  $j$ ).

Observe that

$$\begin{aligned} |\det(C)| &= \left| \sum_{\pi \in S_m} \operatorname{sgn}(\pi) \prod_{1 \leq i \leq m} C_{i\pi(i)} \right| \\ &\leq \sum_{\pi \in S_m} \prod_{1 \leq i \leq m} |C_{i\pi(i)}| \end{aligned}$$

# Bounding the Determinant

Let  $Z$  be the maximum absolute entry occurring in  $\bar{A}$ ,  $\bar{b}$  or  $c$ . Let  $C$  denote the matrix obtained from  $\bar{A}_B$  by replacing the  $j$ -th column with vector  $\bar{b}$  (for some  $j$ ).

Observe that

$$\begin{aligned} |\det(C)| &= \left| \sum_{\pi \in S_m} \operatorname{sgn}(\pi) \prod_{1 \leq i \leq m} C_{i\pi(i)} \right| \\ &\leq \sum_{\pi \in S_m} \prod_{1 \leq i \leq m} |C_{i\pi(i)}| \\ &\leq m! \cdot Z^m . \end{aligned}$$

# Bounding the Determinant

Alternatively, Hadamards inequality gives

$$|\det(C)|$$

# Bounding the Determinant

Alternatively, Hadamard's inequality gives

$$|\det(C)| \leq \prod_{i=1}^m \|C_{*i}\|$$

# Bounding the Determinant

Alternatively, Hadamard's inequality gives

$$|\det(C)| \leq \prod_{i=1}^m \|C_{*i}\| \leq \prod_{i=1}^m (\sqrt{m}Z)$$

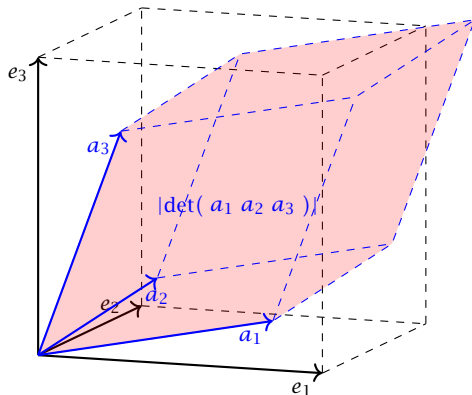
# Bounding the Determinant

Alternatively, Hadamard's inequality gives

$$\begin{aligned} |\det(C)| &\leq \prod_{i=1}^m \|C_{*i}\| \leq \prod_{i=1}^m (\sqrt{m}Z) \\ &\leq m^{m/2} Z^m . \end{aligned}$$



# Hadamards Inequality



Hadamard's inequality says that the volume of the red parallelepiped (**Spat**) is smaller than the volume in the black cube (if  $\|e_1\| = \|a_1\|$ ,  $\|e_2\| = \|a_2\|$ ,  $\|e_3\| = \|a_3\|$ ).

# Ensuring Conditions

Given a **standard minimization LP**

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \end{array}$$

how can we obtain an LP of the required form?

- ▶ **Compute a lower bound on  $c^T x$  for any basic feasible solution.** Add the constraint  $c^T x \geq -dZ(m! \cdot Z^m) - 1$ . **Note that this constraint is superfluous unless the LP is unbounded.**

# Ensuring Conditions

Compute an optimum basis for the new LP.

- ▶ If the cost is  $c^T x = -(dZ)(m! \cdot Z^m) - 1$  we know that the original LP is unbounded.
- ▶ Otw. we have an optimum basis.

In the following we use  $\mathcal{H}$  to denote the set of all constraints apart from the constraint  $c^T x \geq -dZ(m! \cdot Z^m) - 1$ .

We give a routine  $\text{SeidelLP}(\mathcal{H}, d)$  that is given a set  $\mathcal{H}$  of **explicit, non-degenerate** constraints over  $d$  variables, and minimizes  $c^T x$  over all feasible points.

In addition it obeys the implicit constraint  $c^T x \geq -(dZ)(m! \cdot Z^m) - 1$ .

In the following we use  $\mathcal{H}$  to denote the set of all constraints apart from the constraint  $c^T x \geq -dZ(m! \cdot Z^m) - 1$ .

We give a routine  $\text{SeidelLP}(\mathcal{H}, d)$  that is given a set  $\mathcal{H}$  of explicit, non-degenerate constraints over  $d$  variables, and minimizes  $c^T x$  over all feasible points.

In addition it obeys the implicit constraint  $c^T x \geq -(dZ)(m! \cdot Z^m) - 1$ .

In the following we use  $\mathcal{H}$  to denote the set of all constraints apart from the constraint  $c^T x \geq -dZ(m! \cdot Z^m) - 1$ .

We give a routine  $\text{SeidelLP}(\mathcal{H}, d)$  that is given a set  $\mathcal{H}$  of **explicit, non-degenerate** constraints over  $d$  variables, and minimizes  $c^T x$  over all feasible points.

In addition it obeys the implicit constraint  $c^T x \geq -(dZ)(m! \cdot Z^m) - 1$ .

In the following we use  $\mathcal{H}$  to denote the set of all constraints apart from the constraint  $c^T x \geq -dZ(m! \cdot Z^m) - 1$ .

We give a routine  $\text{SeidelLP}(\mathcal{H}, d)$  that is given a set  $\mathcal{H}$  of **explicit, non-degenerate** constraints over  $d$  variables, and minimizes  $c^T x$  over all feasible points.

In addition it obeys the implicit constraint  $c^T x \geq -(dZ)(m! \cdot Z^m) - 1$ .

**Algorithm 1** SeidelLP( $\mathcal{H}, d$ )

---

1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;



**Algorithm 1** SeidelLP( $\mathcal{H}, d$ )

---

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

---

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

---

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

---

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

---

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$
- 6: **if**  $\hat{x}^* = \text{infeasible}$  **then return** infeasible

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

---

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$
- 6: **if**  $\hat{x}^* = \text{infeasible}$  **then return** infeasible
- 7: **if**  $\hat{x}^*$  fulfills  $h$  **then return**  $\hat{x}^*$

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$
- 6: **if**  $\hat{x}^* = \text{infeasible}$  **then return** infeasible
- 7: **if**  $\hat{x}^*$  fulfills  $h$  **then return**  $\hat{x}^*$
- 8: // **optimal solution fulfills  $h$  with equality, i.e.,  $a_h^T x = b_h$**

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$
- 6: **if**  $\hat{x}^* = \text{infeasible}$  **then return** infeasible
- 7: **if**  $\hat{x}^*$  fulfills  $h$  **then return**  $\hat{x}^*$
- 8: // **optimal solution fulfills  $h$  with equality, i.e.,  $a_h^T x = b_h$**
- 9: solve  $a_h^T x = b_h$  for some variable  $x_\ell$ ;
- 10: eliminate  $x_\ell$  in constraints from  $\hat{\mathcal{H}}$  and in implicit constr.;



### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$
- 6: **if**  $\hat{x}^* = \text{infeasible}$  **then return** infeasible
- 7: **if**  $\hat{x}^*$  fulfills  $h$  **then return**  $\hat{x}^*$
- 8: // **optimal solution fulfills  $h$  with equality, i.e.,  $a_h^T x = b_h$**
- 9: solve  $a_h^T x = b_h$  for some variable  $x_\ell$ ;
- 10: eliminate  $x_\ell$  in constraints from  $\hat{\mathcal{H}}$  and in implicit constr.;
- 11:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d - 1)$

### Algorithm 1 SeidelLP( $\mathcal{H}, d$ )

- 1: **if**  $d = 1$  **then** solve 1-dimensional problem and return;
- 2: **if**  $\mathcal{H} = \emptyset$  **then** return  $x$  on implicit constraint hyperplane
- 3: choose **random** constraint  $h \in \mathcal{H}$
- 4:  $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
- 5:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d)$
- 6: **if**  $\hat{x}^* = \text{infeasible}$  **then return** infeasible
- 7: **if**  $\hat{x}^*$  fulfills  $h$  **then return**  $\hat{x}^*$
- 8: // **optimal solution fulfills  $h$  with equality, i.e.,  $a_h^T x = b_h$**
- 9: solve  $a_h^T x = b_h$  for some variable  $x_\ell$ ;
- 10: eliminate  $x_\ell$  in constraints from  $\hat{\mathcal{H}}$  and in implicit constr.;
- 11:  $\hat{x}^* \leftarrow \text{SeidelLP}(\hat{\mathcal{H}}, d - 1)$
- 12: **if**  $\hat{x}^* = \text{infeasible}$  **then**
- 13:     **return** infeasible
- 14: **else**
- 15:     add the value of  $x_\ell$  to  $\hat{x}^*$  and return the solution

## 8 Seidels LP-algorithm

- ▶ If  $d = 1$  we can solve the 1-dimensional problem in time  $\mathcal{O}(\max\{m, 1\})$ .
- ▶ If  $d > 1$  and  $m = 0$  we take time  $\mathcal{O}(d)$  to return  $d$ -dimensional vector  $x$ .
- ▶ The first recursive call takes time  $T(m - 1, d)$  for the call plus  $\mathcal{O}(d)$  for checking whether the solution fulfills  $h$ .
- ▶ If we are unlucky and  $\hat{x}^*$  does not fulfill  $h$  we need time  $\mathcal{O}(d(m + 1)) = \mathcal{O}(dm)$  to eliminate  $x_\ell$ . Then we make a recursive call that takes time  $T(m - 1, d - 1)$ .
- ▶ The probability of being unlucky is at most  $d/m$  as there are at most  $d$  constraints whose removal will decrease the objective function

## 8 Seidels LP-algorithm

- ▶ If  $d = 1$  we can solve the 1-dimensional problem in time  $\mathcal{O}(\max\{m, 1\})$ .
- ▶ If  $d > 1$  and  $m = 0$  we take time  $\mathcal{O}(d)$  to return  $d$ -dimensional vector  $x$ .
- ▶ The first recursive call takes time  $T(m - 1, d)$  for the call plus  $\mathcal{O}(d)$  for checking whether the solution fulfills  $h$ .
- ▶ If we are unlucky and  $\hat{x}^*$  does not fulfill  $h$  we need time  $\mathcal{O}(d(m + 1)) = \mathcal{O}(dm)$  to eliminate  $x_\ell$ . Then we make a recursive call that takes time  $T(m - 1, d - 1)$ .
- ▶ The probability of being unlucky is at most  $d/m$  as there are at most  $d$  constraints whose removal will decrease the objective function

## 8 Seidels LP-algorithm

- ▶ If  $d = 1$  we can solve the 1-dimensional problem in time  $\mathcal{O}(\max\{m, 1\})$ .
- ▶ If  $d > 1$  and  $m = 0$  we take time  $\mathcal{O}(d)$  to return  $d$ -dimensional vector  $x$ .
- ▶ The first recursive call takes time  $T(m - 1, d)$  for the call plus  $\mathcal{O}(d)$  for checking whether the solution fulfills  $h$ .
- ▶ If we are unlucky and  $\hat{x}^*$  does not fulfill  $h$  we need time  $\mathcal{O}(d(m + 1)) = \mathcal{O}(dm)$  to eliminate  $x_\ell$ . Then we make a recursive call that takes time  $T(m - 1, d - 1)$ .
- ▶ The probability of being unlucky is at most  $d/m$  as there are at most  $d$  constraints whose removal will decrease the objective function

## 8 Seidels LP-algorithm

- ▶ If  $d = 1$  we can solve the 1-dimensional problem in time  $\mathcal{O}(\max\{m, 1\})$ .
- ▶ If  $d > 1$  and  $m = 0$  we take time  $\mathcal{O}(d)$  to return  $d$ -dimensional vector  $x$ .
- ▶ The first recursive call takes time  $T(m - 1, d)$  for the call plus  $\mathcal{O}(d)$  for checking whether the solution fulfills  $h$ .
- ▶ If we are unlucky and  $\hat{x}^*$  does not fulfill  $h$  we need time  $\mathcal{O}(d(m + 1)) = \mathcal{O}(dm)$  to eliminate  $x_\ell$ . Then we make a recursive call that takes time  $T(m - 1, d - 1)$ .
- ▶ The probability of being unlucky is at most  $d/m$  as there are at most  $d$  constraints whose removal will decrease the objective function

## 8 Seidels LP-algorithm

- ▶ If  $d = 1$  we can solve the 1-dimensional problem in time  $\mathcal{O}(\max\{m, 1\})$ .
- ▶ If  $d > 1$  and  $m = 0$  we take time  $\mathcal{O}(d)$  to return  $d$ -dimensional vector  $x$ .
- ▶ The first recursive call takes time  $T(m - 1, d)$  for the call plus  $\mathcal{O}(d)$  for checking whether the solution fulfills  $h$ .
- ▶ If we are unlucky and  $\hat{x}^*$  does not fulfill  $h$  we need time  $\mathcal{O}(d(m + 1)) = \mathcal{O}(dm)$  to eliminate  $x_\ell$ . Then we make a recursive call that takes time  $T(m - 1, d - 1)$ .
- ▶ The probability of being unlucky is at most  $d/m$  as there are at most  $d$  constraints whose removal will decrease the objective function

## 8 Seidels LP-algorithm

This gives the recurrence

$$T(m, d) = \begin{cases} \mathcal{O}(\max\{1, m\}) & \text{if } d = 1 \\ \mathcal{O}(d) & \text{if } d > 1 \text{ and } m = 0 \\ \mathcal{O}(d) + T(m - 1, d) + \\ \frac{d}{m}(\mathcal{O}(dm) + T(m - 1, d - 1)) & \text{otw.} \end{cases}$$

Note that  $T(m, d)$  denotes the **expected running time**.



## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

$$T(m, d) = \begin{cases} C \max\{1, m\} & \text{if } d = 1 \\ Cd & \text{if } d > 1 \text{ and } m = 0 \\ Cd + T(m - 1, d) + \\ \frac{d}{m}(Cdm + T(m - 1, d - 1)) & \text{otw.} \end{cases}$$

Note that  $T(m, d)$  denotes the **expected running time**.

## 8 Seidels LP-algorithm

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1)$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\}$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\}$$



## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d)$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq C f(d) \max\{1, m\}$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq C f(d) \max\{1, m\} \text{ for } f(d) \geq d$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq C f(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq C f(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq C f(d) \max\{1, m\} \text{ for } f(d) \geq d$$

**$d > 1; m = 1$  :**

$$T(1, d) = \mathcal{O}(d) + T(0, d) + d(\mathcal{O}(d) + T(0, d - 1))$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq Cf(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq Cf(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq Cf(d) \max\{1, m\} \text{ for } f(d) \geq d$$

**$d > 1; m = 1$  :**

$$\begin{aligned} T(1, d) &= \mathcal{O}(d) + T(0, d) + d(\mathcal{O}(d) + T(0, d-1)) \\ &\leq Cd + Cd + Cd^2 + dCf(d-1) \end{aligned}$$

## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq Cf(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq Cf(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq Cf(d) \max\{1, m\} \text{ for } f(d) \geq d$$

**$d > 1; m = 1$  :**

$$\begin{aligned} T(1, d) &= \mathcal{O}(d) + T(0, d) + d(\mathcal{O}(d) + T(0, d-1)) \\ &\leq Cd + Cd + Cd^2 + dCf(d-1) \\ &\leq Cf(d) \max\{1, m\} \end{aligned}$$



## 8 Seidels LP-algorithm

Let  $C$  be the largest constant in the  $\mathcal{O}$ -notations.

We show  $T(m, d) \leq Cf(d) \max\{1, m\}$ .

**$d = 1$ :**

$$T(m, 1) \leq C \max\{1, m\} \leq Cf(1) \max\{1, m\} \text{ for } f(1) \geq 1$$

**$d > 1; m = 0$  :**

$$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq Cf(d) \max\{1, m\} \text{ for } f(d) \geq d$$

**$d > 1; m = 1$  :**

$$\begin{aligned} T(1, d) &= \mathcal{O}(d) + T(0, d) + d(\mathcal{O}(d) + T(0, d-1)) \\ &\leq Cd + Cd + Cd^2 + dCf(d-1) \\ &\leq Cf(d) \max\{1, m\} \text{ for } f(d) \geq 3d^2 + df(d-1) \end{aligned}$$

## 8 Seidels LP-algorithm

**$d > 1; m > 1 :$**

(by induction hypothesis statm. true for  $d' < d, m' \geq 0;$

and for  $d' = d, m' < m$ )

## 8 Seidels LP-algorithm

**$d > 1; m > 1 :$**

(by induction hypothesis statm. true for  $d' < d, m' \geq 0;$

and for  $d' = d, m' < m$ )

$$T(m, d) = \mathcal{O}(d) + T(m - 1, d) + \frac{d}{m} \left( \mathcal{O}(dm) + T(m - 1, d - 1) \right)$$

## 8 Seidels LP-algorithm

$d > 1; m > 1 :$

(by induction hypothesis statm. true for  $d' < d, m' \geq 0$ ;

and for  $d' = d, m' < m$ )

$$\begin{aligned} T(m, d) &= \mathcal{O}(d) + T(m - 1, d) + \frac{d}{m} \left( \mathcal{O}(dm) + T(m - 1, d - 1) \right) \\ &\leq Cd + Cf(d)(m - 1) + Cd^2 + \frac{d}{m} Cf(d - 1)(m - 1) \end{aligned}$$

## 8 Seidels LP-algorithm

**$d > 1; m > 1 :$**

(by induction hypothesis statm. true for  $d' < d, m' \geq 0$ ;

and for  $d' = d, m' < m$ )

$$\begin{aligned}T(m, d) &= \mathcal{O}(d) + T(m - 1, d) + \frac{d}{m} \left( \mathcal{O}(dm) + T(m - 1, d - 1) \right) \\&\leq Cd + Cf(d)(m - 1) + Cd^2 + \frac{d}{m} Cf(d - 1)(m - 1) \\&\leq 2Cd^2 + Cf(d)(m - 1) + dCf(d - 1)\end{aligned}$$

## 8 Seidels LP-algorithm

$d > 1; m > 1 :$

(by induction hypothesis statm. true for  $d' < d, m' \geq 0$ ;

and for  $d' = d, m' < m$ )

$$\begin{aligned}T(m, d) &= \mathcal{O}(d) + T(m - 1, d) + \frac{d}{m} \left( \mathcal{O}(dm) + T(m - 1, d - 1) \right) \\&\leq Cd + Cf(d)(m - 1) + Cd^2 + \frac{d}{m} Cf(d - 1)(m - 1) \\&\leq 2Cd^2 + Cf(d)(m - 1) + dCf(d - 1) \\&\leq Cf(d)m\end{aligned}$$

## 8 Seidels LP-algorithm

$d > 1; m > 1 :$

(by induction hypothesis statm. true for  $d' < d, m' \geq 0$ ;

and for  $d' = d, m' < m$ )

$$\begin{aligned}T(m, d) &= \mathcal{O}(d) + T(m - 1, d) + \frac{d}{m} \left( \mathcal{O}(dm) + T(m - 1, d - 1) \right) \\&\leq Cd + Cf(d)(m - 1) + Cd^2 + \frac{d}{m} Cf(d - 1)(m - 1) \\&\leq 2Cd^2 + Cf(d)(m - 1) + dCf(d - 1) \\&\leq Cf(d)m\end{aligned}$$

if  $f(d) \geq df(d - 1) + 2d^2$ .

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .



## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$f(d)$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$f(d) = 3d^2 + df(d-1)$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$\begin{aligned}f(d) &= 3d^2 + df(d-1) \\ &= 3d^2 + d \left[ 3(d-1)^2 + (d-1)f(d-2) \right]\end{aligned}$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$\begin{aligned}f(d) &= 3d^2 + df(d-1) \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1)f(d-2) \right] \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1) \left[ 3(d-2)^2 + (d-2)f(d-3) \right] \right]\end{aligned}$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$\begin{aligned}f(d) &= 3d^2 + df(d-1) \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1)f(d-2) \right] \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1) \left[ 3(d-2)^2 + (d-2)f(d-3) \right] \right] \\&= 3d^2 + 3d(d-1)^2 + 3d(d-1)(d-2)^2 + \dots \\&\quad + 3d(d-1)(d-2) \cdot \dots \cdot 4 \cdot 3 \cdot 2 \cdot 1^2\end{aligned}$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$\begin{aligned}f(d) &= 3d^2 + df(d-1) \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1)f(d-2) \right] \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1) \left[ 3(d-2)^2 + (d-2)f(d-3) \right] \right] \\&= 3d^2 + 3d(d-1)^2 + 3d(d-1)(d-2)^2 + \dots \\&\quad + 3d(d-1)(d-2) \cdot \dots \cdot 4 \cdot 3 \cdot 2 \cdot 1^2 \\&= 3d! \left( \frac{d^2}{d!} + \frac{(d-1)^2}{(d-1)!} + \frac{(d-2)^2}{(d-2)!} + \dots \right)\end{aligned}$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$\begin{aligned}f(d) &= 3d^2 + df(d-1) \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1)f(d-2) \right] \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1) \left[ 3(d-2)^2 + (d-2)f(d-3) \right] \right] \\&= 3d^2 + 3d(d-1)^2 + 3d(d-1)(d-2)^2 + \dots \\&\quad + 3d(d-1)(d-2) \cdot \dots \cdot 4 \cdot 3 \cdot 2 \cdot 1^2 \\&= 3d! \left( \frac{d^2}{d!} + \frac{(d-1)^2}{(d-1)!} + \frac{(d-2)^2}{(d-2)!} + \dots \right) \\&= \mathcal{O}(d!)\end{aligned}$$

## 8 Seidels LP-algorithm

- ▶ Define  $f(1) = 3 \cdot 1^2$  and  $f(d) = df(d-1) + 3d^2$  for  $d > 1$ .

Then

$$\begin{aligned}f(d) &= 3d^2 + df(d-1) \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1)f(d-2) \right] \\&= 3d^2 + d \left[ 3(d-1)^2 + (d-1) \left[ 3(d-2)^2 + (d-2)f(d-3) \right] \right] \\&= 3d^2 + 3d(d-1)^2 + 3d(d-1)(d-2)^2 + \dots \\&\quad + 3d(d-1)(d-2) \cdot \dots \cdot 4 \cdot 3 \cdot 2 \cdot 1^2 \\&= 3d! \left( \frac{d^2}{d!} + \frac{(d-1)^2}{(d-1)!} + \frac{(d-2)^2}{(d-2)!} + \dots \right) \\&= \mathcal{O}(d!)\end{aligned}$$

since  $\sum_{i \geq 1} \frac{i^2}{i!}$  is a constant.