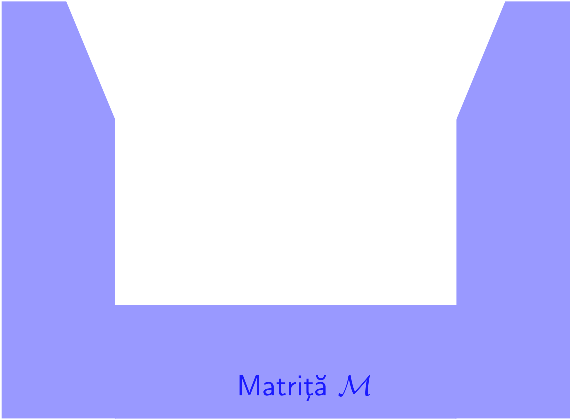
Elemente de programare liniară

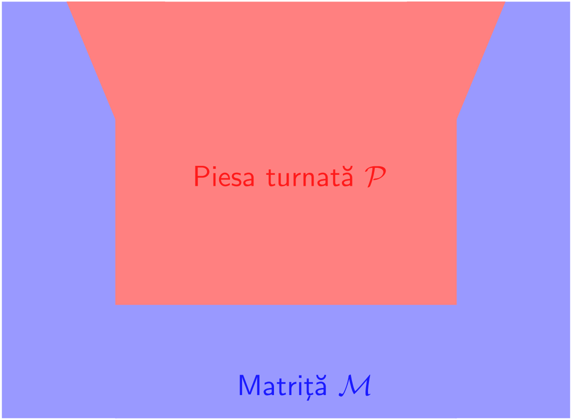
Mihai-Sorin Stupariu

Sem. I, 2016-2017

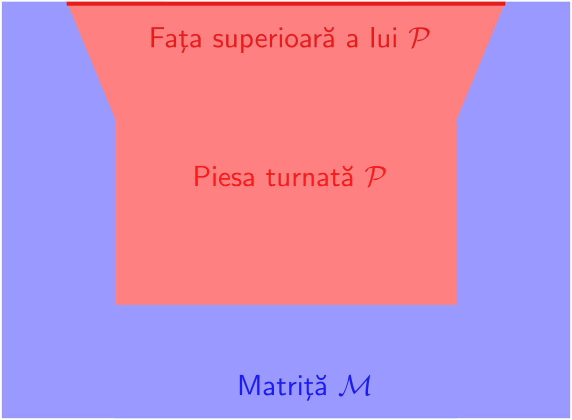
Turnarea pieselor în matrițe



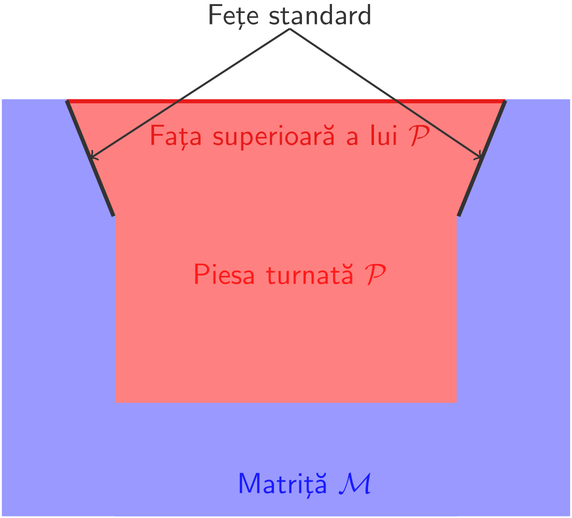
Turnarea pieselor în matrițe



Turnarea pieselor în matrițe



Turnarea pieselor în matrițe



Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.

Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.

Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?

Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- **Convenții.**

Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- **Convenții.**
  - Obiectele: **poliedrale**.

Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- **Convenții.**
  - Obiectele: **poliedrale**.
  - Matrițele: formate dintr-o singură piesă; fiecărui obiect $\mathcal{P}$ îi este asociată o matriță $\mathcal{M}_{\mathcal{P}}$

Problematizare

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- **Convenții.**
  - Obiectele: **poliedrale**.
  - Matrițele: formate dintr-o singură piesă; fiecărui obiect $\mathcal{P}$ îi este asociată o matriță $\mathcal{M}_{\mathcal{P}}$
  - Obiectul: extras printr-o singură translație (sau o succesiune de translații)

Terminologie și convenții

- **Alegerea orientării:** diverse orientări ale obiectului pot generа diverse matrițe.

Terminologie și convenții

- **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- **Fața superioară:** prin convenție, obiectele au (cel puțin) o fața superioară (este orizontală, este singura care nu este adiacentă cu matrița). Celelalte fețe: **standard**; orice față standard f a obiectului corespunde unei fețe $\hat{f}$ a matriței.

Terminologie și convenții

- **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- **Fața superioară:** prin convenție, obiectele au (cel puțin) o fața superioară (este orizontală, este singura care nu este adiacentă cu matrița). Celelalte fețe: **standard**; orice față standard f a obiectului corespunde unei fețe $\hat{f}$ a matriței.
- **Obiect care poate fi turnat (*castable*):** există o orientare pentru care acesta poate fi turnat și apoi extras printr-o translație (succesiune de translații): *direcție admisibilă*.

Terminologie și convenții

- **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- **Fața superioară:** prin convenție, obiectele au (cel puțin) o fața superioară (este orizontală, este singura care nu este adiacentă cu matrița). Celelalte fețe: **standard**; orice față standard f a obiectului corespunde unei fețe $\hat{f}$ a matriței.
- **Obiect care poate fi turnat (*castable*):** există o orientare pentru care acesta poate fi turnat și apoi extras printr-o translație (succesiune de translații): *direcție admisibilă*.
- **Convenții:** Matrița este paralelipipedică și are o cavitate corespunzătoare obiectului; fața superioară a obiectului (și a matriței) este perpendiculară cu planul Oxy.

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă
- **În general:** o față $\hat{f}$ a matriței pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și $\vec{d}$ este mai mic de 90° împiedică translația în direcția $\vec{d}$

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă
- **În general:** o față $\hat{f}$ a matriței pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și $\vec{d}$ este mai mic de 90° împiedică translația în direcția $\vec{d}$
- **Propoziție.** *Un poliedru $\mathcal{P}$ poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția $\vec{d}$ dacă și numai dacă $\vec{d}$ face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui $\mathcal{P}$.*

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă
- **În general:** o față $\hat{f}$ a matriței pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și $\vec{d}$ este mai mic de 90° împiedică translația în direcția $\vec{d}$
- **Propoziție.** *Un poliedru $\mathcal{P}$ poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția $\vec{d}$ dacă și numai dacă $\vec{d}$ face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui $\mathcal{P}$.*
- **Reformulare.** Dat $\mathcal{P}$, trebuie găsită o direcție $\vec{d}$ astfel încât, pentru fiecare față standard f, unghiul dintre $\vec{d}$ și $\vec{\nu}(f)$ să fie cel puțin 90°.

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă
- **În general:** o față $\hat{f}$ a matriței pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și $\vec{d}$ este mai mic de 90° împiedică translația în direcția $\vec{d}$
- **Propoziție.** *Un poliedru $\mathcal{P}$ poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția $\vec{d}$ dacă și numai dacă $\vec{d}$ face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui $\mathcal{P}$.*
- **Reformulare.** Dat $\mathcal{P}$, trebuie găsită o direcție $\vec{d}$ astfel încât, pentru fiecare față standard f, unghiul dintre $\vec{d}$ și $\vec{\nu}(f)$ să fie cel puțin 90°.
- **Analitic:** fiecare față definește un semiplan

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă
- **În general:** o față $\hat{f}$ a matriței pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și $\vec{d}$ este mai mic de 90° împiedică translația în direcția $\vec{d}$
- **Propoziție.** *Un poliedru $\mathcal{P}$ poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția $\vec{d}$ dacă și numai dacă $\vec{d}$ face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui $\mathcal{P}$.*
- **Reformulare.** Dat $\mathcal{P}$, trebuie găsită o direcție $\vec{d}$ astfel încât, pentru fiecare față standard f, unghiul dintre $\vec{d}$ și $\vec{\nu}(f)$ să fie cel puțin 90°.
- **Analitic:** fiecare față definește un semiplan
- **Concluzie:** Fie $\mathcal{P}$ un poliedru. Mulțimea direcțiilor admisibile este dată de o intersecție de semiplane.

Fundamente geometrice

- **Condiție necesară:** direcția de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă
- **În general:** o față $\hat{f}$ a matriței pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și $\vec{d}$ este mai mic de 90° împiedică translația în direcția $\vec{d}$
- **Propoziție.** *Un poliedru $\mathcal{P}$ poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția $\vec{d}$ dacă și numai dacă $\vec{d}$ face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui $\mathcal{P}$.*
- **Reformulare.** Dat $\mathcal{P}$, trebuie găsită o direcție $\vec{d}$ astfel încât, pentru fiecare față standard f, unghiul dintre $\vec{d}$ și $\vec{\nu}(f)$ să fie cel puțin 90°.
- **Analitic:** fiecare față definește un semiplan
- **Concluzie:** Fie $\mathcal{P}$ un poliedru. Mulțimea direcțiilor admisibile este dată de o intersecție de semiplane.
- **Teoremă.** *Fie $\mathcal{P}$ un poliedru cu n fețe. Se poate decide dacă $\mathcal{P}$ reprezintă un obiect care poate fi turnat în $O(n^2)$ timp și folosind $O(n)$ spațiu. În caz afirmativ, o matriță și o direcție admisibiă în care poate fi extras $\mathcal{P}$ este determinată cu aceeași complexitate timp.*

Formularea problemei

- Fie $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$ o mulțime de semiplane din $\mathbb{R}^2$; semiplanul H_i dat de o relație de forma

$$a_i x + b_i y \leq c_i$$

Formularea problemei

- Fie $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$ o mulțime de semiplane din $\mathbb{R}^2$; semiplanul H_i dat de o relație de forma

$$a_i x + b_i y \leq c_i$$

- Intersecția $H_1 \cap H_2 \cap \ldots \cap H_n$ este dată de un sistem de inecuații; este o mulțime poligonală convexă, mărginită de cel mult n muchii (poate fi vidă, mărginită, nemărginită,...)

Algoritm IntersectiiSemiplane ($\mathcal{H}$)

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$

Algoritm IntersectiiSemiplane ($\mathcal{H}$)

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

Algoritm $\textsc{IntersectiiSemiplane}$ $(\mathcal{H})$

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. 1. **if** $\mathrm{card}(\mathcal{H}) = 1$

Algoritm $\textsc{IntersectiiSemiplane}$ ($\mathcal{H}$)

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. 1. **if** $\mathrm{card}(\mathcal{H}) = 1$
2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$

Algoritm IntersectiiSemiplane $(\mathcal{H})$

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. 1. **if** $\mathrm{card}(\mathcal{H}) = 1$
2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
3. **else** descompune $\mathcal{H}$ în două mulțimi $\mathcal{H}_1$, $\mathcal{H}_2$ având fiecare $[n/2]$ elemente

Algoritm $\textsc{IntersectiiSemiplane}$ $(\mathcal{H})$

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. 1. **if** $\mathrm{card}(\mathcal{H}) = 1$
2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
3. **else** descompune $\mathcal{H}$ în două mulțimi $\mathcal{H}_1$, $\mathcal{H}_2$ având fiecare $[n/2]$ elemente
4. $\mathcal{C}_1 \leftarrow \textsc{IntersectiiSemiplane}$ $(\mathcal{H}_1)$

Algoritm IntersectiiSemiplane ($\mathcal{H}$)

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. **if** $\mathrm{card}(\mathcal{H}) = 1$
    then $\mathcal{C} \leftarrow H \in \mathcal{H}$
    else descompune $\mathcal{H}$ în două mulțimi $\mathcal{H}_1$, $\mathcal{H}_2$ având fiecare $[n/2]$ elemente
        $\mathcal{C}_1 \leftarrow$ IntersectiiSemiplane ($\mathcal{H}_1$)
        $\mathcal{C}_2 \leftarrow$ IntersectiiSemiplane ($\mathcal{H}_2$)

Algoritm IntersectiiSemiplane ($\mathcal{H}$)

- **Input.** O mulțime $\mathcal{H}$ de semiplane din planul $\mathbb{R}^2$
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. 1. **if** $\mathrm{card}(\mathcal{H}) = 1$
2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
3. **else** descompune $\mathcal{H}$ în două mulțimi $\mathcal{H}_1$, $\mathcal{H}_2$ având fiecare $[n/2]$ elemente
4. $\mathcal{C}_1 \leftarrow$ IntersectiiSemiplane ($\mathcal{H}_1$)
5. $\mathcal{C}_2 \leftarrow$ IntersectiiSemiplane ($\mathcal{H}_2$)
6. $\mathcal{C} \leftarrow$ IntersecteazaRegiuniConvexe ($\mathcal{C}_1, \mathcal{C}_2$)

Rezultate principale

- **Propoziție.** *Aplicând direct algoritmii de overlay, intersecția dintre două regiuni convexe* (IntersecteazaRegiuniConvexe) *poate fi calculată cu complexitate-timp* $O(n \log n)$*; în particular algoritmul* IntersectiiSemiplane *are complexitate* $O(n \log^2 n)$.

Rezultate principale

- **Propoziție.** *Aplicând direct algoritmii de overlay, intersecția dintre două regiuni convexe (*INTERSECTEAZAREGIUNICONVEXE*) poate fi calculată cu complexitate-timp* $O(n \log n)$*; în particular algoritmul* INTERSECTIISEMIPLANE *are complexitate* $O(n \log^2 n)$.
- **Teoremă.** *Algoritmul* INTERSECTEAZAREGIUNICONVEXE *poate fi îmbunătățit, astfel încât complexitatea-timp să fie liniară.*

Rezultate principale

- **Propoziție.** *Aplicând direct algoritmii de overlay, intersecția dintre două regiuni convexe (*IntersecteazaRegiuniConvexe*) poate fi calculată cu complexitate-timp* $O(n \log n)$*; în particular algoritmul* IntersectiiSemiplane *are complexitate* $O(n \log^2 n)$.
- **Teoremă.** *Algoritmul* IntersecteazaRegiuniConvexe *poate fi îmbunătățit, astfel încât complexitatea-timp să fie liniară.*
- **Teoremă.** *Intersecția unei mulțimi de* n *semiplane poate fi determinată cu complexitate-timp* $O(n \log n)$ *și folosind* $O(n)$ *memorie.*

Problematizare

- **Formulare generală (în spațiul *d*-dimensional):**

$$\text{maximizează}(c_1x_1 + c_2x_2 + \ldots + c_dx_d)$$

date constrângerile liniare (inegalități)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2d}x_d \leq b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nd}x_d \leq b_n \end{cases}$$

Problematizare

- **Formulare generală (în spațiul *d*-dimensional):**

$$\text{maximizează}(c_1x_1 + c_2x_2 + \ldots + c_dx_d)$$

date constrângerile liniare (inegalități)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2d}x_d \leq b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nd}x_d \leq b_n \end{cases}$$

- **Terminologie:** date de intrare, funcție obiectiv, constrângeri, regiune realizabilă (fezabilă)

Problematizare

- **Formulare generală (în spațiul d-dimensional):**

$$\text{maximizează}(c_1x_1 + c_2x_2 + \ldots + c_dx_d)$$

date constrângerile liniare (inegalități)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2d}x_d \leq b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nd}x_d \leq b_n \end{cases}$$

- **Terminologie:** date de intrare, funcție obiectiv, constrângeri, regiune realizabilă (fezabilă)
- **Exemple:** probleme de programare liniară 1-dimensională, 2-dimensională.

Rezultate

- **Lemă.** (Pentru $d = 1$) *Un program liniar* 1*-dimensional poate fi rezolvat în timp liniar.*

Rezultate

- **Lemă.** (Pentru $d = 1$) *Un program liniar* 1*-dimensional poate fi rezolvat în timp liniar.*
- **Interpretare a cerinței de maximizare:** Maximizarea funcției obiectiv revine la a determina un punct al cărui vector de poziție are proiecția maximă de direcția dată de vectorul $\vec{c} = (c_1, c_2, \ldots, c_d)$.

Rezultate

- **Lemă.** (Pentru $d = 1$) *Un program liniar* 1*-dimensional poate fi rezolvat în timp liniar.*
- **Interpretare a cerinței de maximizare:** Maximizarea funcției obiectiv revine la a determina un punct al cărui vector de poziție are proiecția maximă de direcția dată de vectorul $\vec{c} = (c_1, c_2, \ldots, c_d)$.
- Pentru o problemă de programare liniară în plan ($d = 2$) pot fi distinse patru situații: (i) o soluție unică; (ii) toate punctele de pe o muchie sunt soluții; (iii) regiunea fezabilă este nemărginită și pot fi găsite soluții de-a lungul unei semidrepte; (iv) regiunea fezabilă este vidă.

Algoritm LPMARG2D $(H, \overrightarrow{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ din $\mathbb{R}^2$

Algoritm $\mathrm{LPMARG2D}$ $(H, \overrightarrow{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\overrightarrow{c}}(p)$.

Algoritm $\mathrm{LPMARG2D}\ (H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0

Algoritm LPMARG2D $(H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ semiplanele din H

Algoritm $\mathrm{LPMARG2D}\ (H, \overrightarrow{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\overrightarrow{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ semiplanele din H
3. **for** $i \leftarrow 1$ **to** n

Algoritm LPMARG2D $(H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ semiplanele din H
3. **for** $i \leftarrow 1$ **to** n
4. **do if** $v_{i-1} \in h_i$

Algoritm LPMARG2D $(H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ semiplanele din H
3. **for** $i \leftarrow 1$ **to** n
4.   **do if** $v_{i-1} \in h_i$
5.   **then** $v_i \leftarrow v_{i-1}$

Algoritm LPMARG2D $(H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ semiplanele din H
3. **for** $i \leftarrow 1$ **to** n
4.   **do if** $v_{i-1} \in h_i$
5.   **then** $v_i \leftarrow v_{i-1}$
6.   **else** $v_i \leftarrow$ punctul p de pe d_i care maximizează $f_{\vec{c}}(p)$ date constrângerile din H_i

Algoritm LPMARG2D $(H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

$v_0 \leftarrow$ "colțul" lui c_0

fie $h_1, h_2, \dots, h_n$ semiplanele din H

for $i \leftarrow 1$ **to** n

    do if $v_{i-1} \in h_i$

        then $v_i \leftarrow v_{i-1}$

        else $v_i \leftarrow$ punctul p de pe d_i care maximizează $f_{\vec{c}}(p)$ date constrângerile din H_i

            if p nu există

Algoritm LPMARG2D $(H, \vec{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \vec{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \vec{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.

$v_0 \leftarrow$ "colțul" lui c_0
fie $h_1, h_2, \ldots, h_n$ semiplanele din H
for $i \leftarrow 1$ **to** n
  do if $v_{i-1} \in h_i$
    then $v_i \leftarrow v_{i-1}$
    else $v_i \leftarrow$ punctul p de pe d_i care
      maximizează $f_{\vec{c}}(p)$ date constrângerile din H_i
      if p nu există
        then raportează "nefezabil" **end**

Algoritm LPMARG2D $(H, \overrightarrow{c}, m_1, m_2)$

- **Input.** Un program liniar $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ din $\mathbb{R}^2$
- **Output.** Dacă $(H \cup \{m_1, m_2\}, \overrightarrow{c})$ nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\overrightarrow{c}}(p)$.

1. $v_0 \leftarrow$ "colțul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ semiplanele din H
3. **for** $i \leftarrow 1$ **to** n
4.   **do if** $v_{i-1} \in h_i$
5.   **then** $v_i \leftarrow v_{i-1}$
6.   **else** $v_i \leftarrow$ punctul p de pe d_i care maximizează $f_{\overrightarrow{c}}(p)$ date constrângerile din H_i
7.   **if** p nu există
8.   **then** raportează "nefezabil" **end**
9. **return** v_n

Algoritm aleatoriu

- Pasul **2.** este înlocuit cu:
  2'. Calculează o permutare arbitrară a semiplanelor, folosind o procedură adecvată.

Algoritm aleatoriu

- Pasul **2.** este înlocuit cu:
  2'. Calculează o permutare arbitrară a semiplanelor, folosind o procedură adecvată.
- Algoritmul incremental $\mathrm{LPMARG2D}$ are complexitate-timp $O(n^2)$, iar varianta bazată pe alegerea aleatorie a semiplanelor are complexitate-timp medie $O(n)$ (n este numărul semiplanelor).