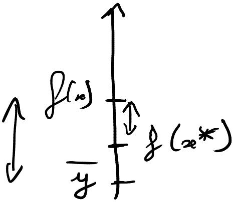
5. Heuristiques et programmation linéaire

8 Novembre 2023

Heuristique



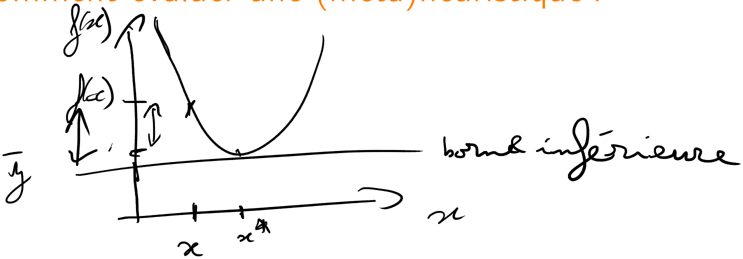
$$(P):\ \min_{x \in \mathcal{X}} f(x)$$

$\mathcal{X}$ ensemble fini, trop grand pour énumérer toutes les solutions en temps raisonnable

Heuristique. Algorithme spécifique au problème considéré qui renvoie une "bonne" solution

- Pas de garantie d'optimalité
- Donne une borne supérieure
- Rapide
- Pas de méthode générale pour les concevoir

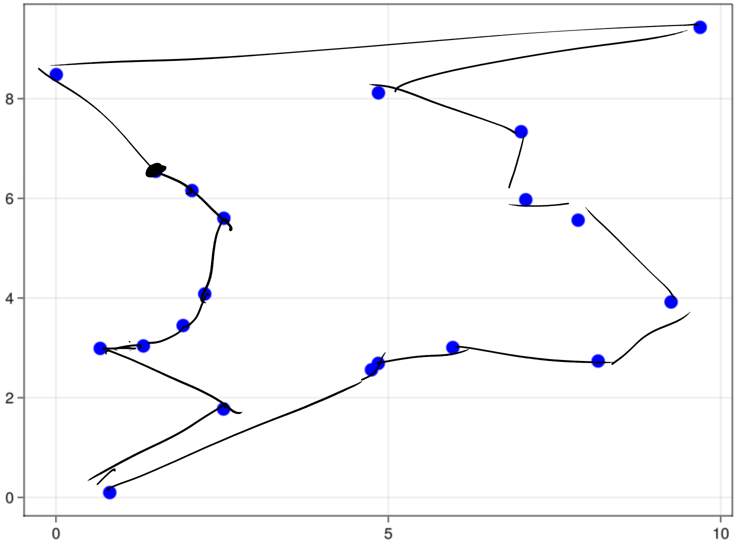
Comment évaluer une (méta)heuristique ?



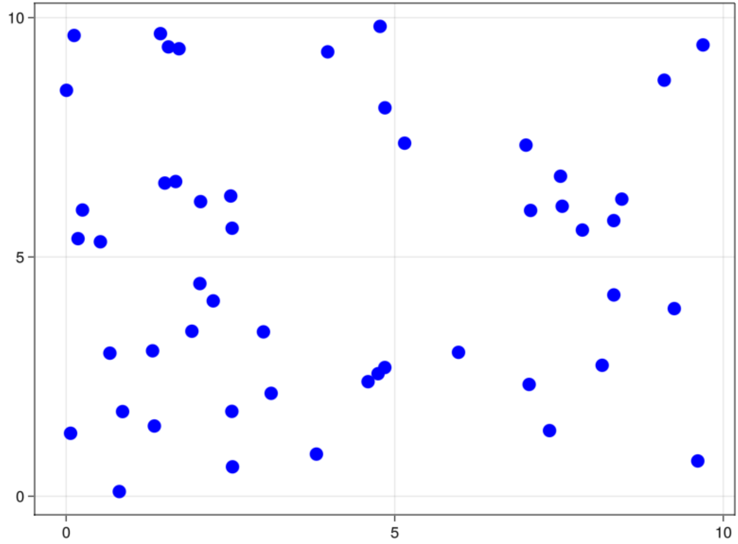
Comment évaluer une (méta)heuristique ?

- Calcul de bornes inférieures
- Expérimentation sur des familles d'instance

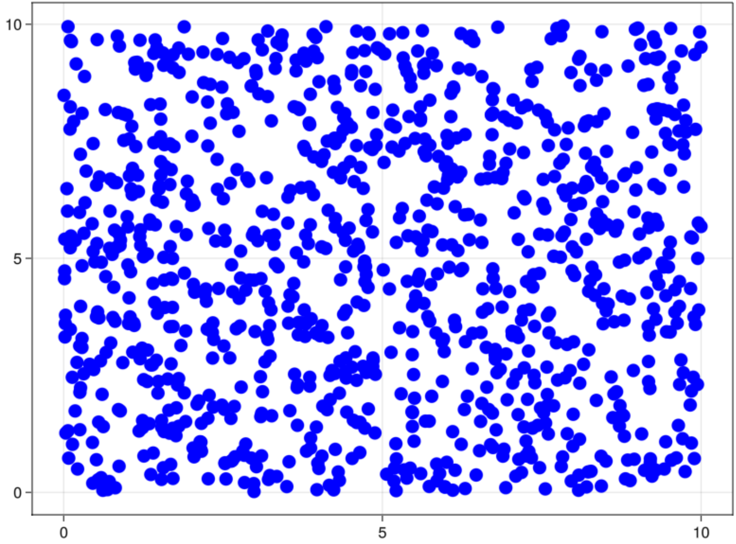
Exemple : problème du voyageur de commerce 20 villes



Exemple : problème du voyageur de commerce 50 villes

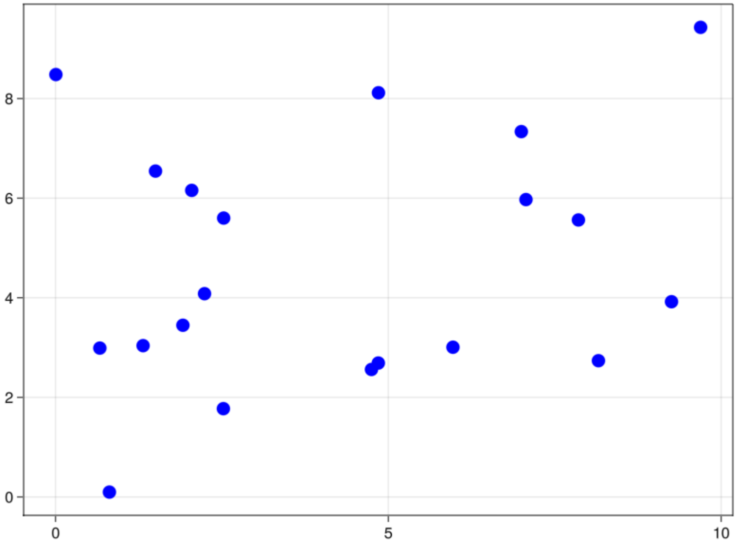


Exemple : problème du voyageur de commerce 1000 villes

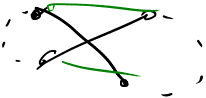


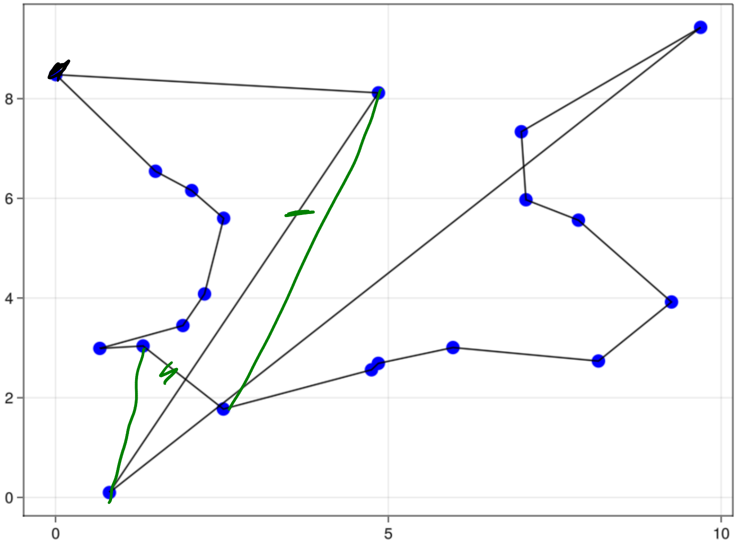
Heuristique constructive

Heuristique constructive : plus proche voisin



Heuristique constructive : plus proche voisin





Métaheuristiques

Métaheuristique. Méthode générique qui peut s'appliquer à la plupart des problèmes.

Deux grandes familles :

- Recherche locale
- Métaheuristiques à population

Recherche locale

1. Initialisation : solution réalisable de départ (via une heuristique par exemple)
2. On essaie d'améliorer la solution courante via des modifications locales $\Longrightarrow$ voisinage

Problème : minimum local

Exemple : voisinages pour le TSP (N villes)

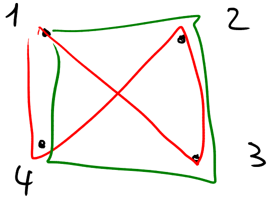
Encodage d'une solution ?

- permutation des villes $[\![1, N]\!]$ → ordre du tour
- matrice $M \in \mathbb{R}^{N \times N}$

$$M_{ij} = \begin{cases} 1 & \text{si } (i,j) \in \text{tour} \\ 0 & \text{sinon} \end{cases}$$

Exemple : voisinages pour le TSP

Encodage d'une solution ? permutation de $\{1, \ldots, n\}$ = T



Voisinages :

- Swap :
  ↳ tire aléatoirement i, j
  ↳ on échange T_i et T_j

ex : $\underset{T}{[1, \underset{i}{3}, \underset{j}{2}, 4]} \longrightarrow \underset{T'}{[1, 2, 3, 4]}$

- 2 - opt

Exemple : voisinages pour le TSP

Encodage d'une solution ? permutation de $\{1, \ldots, n\}$

Voisinages :

1. Echange de deux villes dans la permutation

Exemple : voisinages pour le TSP

Encodage d'une solution ? permutation de $\{1, \ldots, n\}$

Voisinages :

1. Echange de deux villes dans la permutation
2. 2-opt, k-opt

Méthodes avancées

- Descente à voisinages variables $\rightarrow$ alterne entre différents voisinages $V_1, \ldots, V_n$
- Descente à voisinages larges $\rightarrow$ PLNE pour trouver la meilleure solution du voisinage

1. Heuristiques et métaheuristiques
   - Métaheuristiques
     - Recherche locale
     - Recuit simulé
     - Recherche avec tabou
     - Algorithmes évolutionnaires
   - Comment aborder un problème avec des heuristiques ?
2. Programme linéaire
   - Algorithme du simplexe
   - Dualité
   - Totale unimodularité

Recuit simulé

Même fonctionnement que la recherche locale, sauf qu'on accepte certains changements $x \to x'$ non améliorants, avec probabilité

$$p = \exp\left(-\frac{\overbrace{f(x') - f(x)}^{\Delta f}}{T_i}\right) \quad \text{si } \Delta f > 0$$

- n itérations par palier de température i
- p diminiue à chaque changement de palier : $T_{i+1} \leftarrow \alpha T_i$
- Condition d'arrêt : $T_k < T_{min}$

$\alpha < 1$

Paramétrisation

Hyperparamètres à calibrer :

1. Solution initiale x_0
2. Température initiale T_0
3. Température finale T_{min}
4. Vitesse de diminution de la température α (entre 0.85 et 0.95 en pratique)
5. Nombre d'itérations n par palier

Choix de T_0 :

$$T_0 = -\frac{\Delta_0}{\ln p_0}$$

$$p_0 = \exp\left(\frac{-\Delta_0}{T_0}\right)$$

1. Heuristiques et métaheuristiques
   - Métaheuristiques
     - Recherche locale
     - Recuit simulé
     - Recherche avec tabou
     - Algorithmes évolutionnaires
   - Comment aborder un problème avec des heuristiques ?

2. Programme linéaire
   - Algorithme du simplexe
   - Dualité
   - Totale unimodularité

Recherche tabou

Maintien en mémoire d'une liste de solutions "tabou" déjà visitées.

1. Heuristiques et métaheuristiques
   - Métaheuristiques
     - Recherche locale
     - Recuit simulé
     - Recherche avec tabou
     - Algorithmes évolutionnaires
   - Comment aborder un problème avec des heuristiques ?

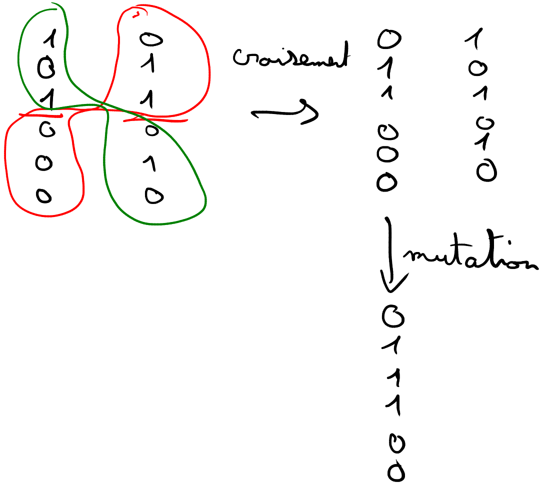
2. Programme linéaire
   - Algorithme du simplexe
   - Dualité
   - Totale unimodularité

Algorithmes évolutionnaires

- croisement
- mutation
- sélection

- Solution réalisable = chromosome.
- On maintient une population de chromosomes.
- On autorise les chromosomes à se croiser avec une probabilité d'autant plus grande que les chromosomes sont de qualité.
- Les chromosomes peuvent muter.
- Les plus mauvais chromosomes meurent, laissant la taille de la population constante

Crucial : codage d'une solution réalisable sur un chromosome, nature du croisement (qui doit maintenir les avantages compétitifs de ses parents).


1 0 1 0 0 0
0 1 1 0 1 0
croisement
0 1 1 0 0 0
1 0 1 0 1 0
mutation
0 1 1 1 0 0

Comment gérer les contraintes ?

- **Rejet** : on accepte pas les solutions non réalisables
- **Pénalisation** : forte pénalisation de l'objectif en fonction de la violation des contraintes
- **Réparation** : heuristique réparant une solution non admissible en solution admissible

Quand la poésie remplace les idées scientifiques

Des centaines de métaheuristiques, toutes plus “novel” et “efficient” que les autres (sur des petites instances faciles ont été présentées dans la littérature, dans une course à la métaphore).
Quelle métaphore ci-dessous n'est pas une métaheuristique publiée ?

- bees optimization
- harmony search orchestra
- ants
- football team
- flies, fireflies
- colonizing behavior of empires
- leaping frog
- cuckoo search
- termites
- water drops
- honey bees
- sperm cells moving to fertilize an egg
- spiraling movements of galaxies
- bats

Quand la poésie remplace les idées scientifiques

Des centaines de métaheuristiques, toutes plus “novel” et “efficient” que les autres (sur des petites instances faciles ont été présentées dans la littérature, dans une course à la métaphore).
Quelle métaphore ci-dessous n’est pas une métaheuristique publiée ?

- bees optimization
- harmony search orchestra
- ants
- football team
- flies, fireflies
- colonizing behavior of empires
- leaping frog
- cuckoo search
- termites
- water drops
- honey bees
- sperm cells moving to fertilize an egg
- spiraling movements of galaxies
- bats

$\Longrightarrow$ Pas d’intrus
(c.f. *Metaheuristics the metaphor exposed, Sörensen 2013*)

1. Heuristiques et métaheuristiques
   - Métaheuristiques
     - Recherche locale
     - Recuit simulé
     - Recherche avec tabou
     - Algorithmes évolutionnaires
   - Comment aborder un problème avec des heuristiques ?

2. Programme linéaire
   - Algorithme du simplexe
   - Dualité
   - Totale unimodularité

Le KIRO

1. Choix d'un langage de programmation : Julia, C++, (Python)
2. Heuristique constructive
3. Recherche locale (définir un encodage et voisinage intéressant) initialisée avec le glouton
4. (Recuit simulé)

Le projet

1. Calcul de bornes inférieures pour estimer la qualité des solutions
2. Génération d'un jeu d'instances
3. Métaheuristiques avancées

Formes inéquationnelle, canonique, standard

① Forme inéquationnelle : $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$

$$\begin{cases} \min_x & c^T x \\ \text{s.c.} & Ax \leq b \\ & x \in \mathbb{R}^n \end{cases}$$

② Forme canonique

$$\begin{cases} \min & c^T x \\ \text{s.c.} & Ax \leq b \\ & x \geq 0 \end{cases}$$

③ Forme standard

$$\begin{cases} \min & c^T x \\ \text{s.c.} & Ax = b \\ & x \geq 0 \end{cases}$$

① => ② $x' = [u, v] \in \mathbb{R}_+^{2n}$

$c' = [c \ \ -c]$ $b' = b$

$A' = (A \ \ -A)$

$$\begin{cases} \min & c^T u - c^T v \\ \text{s.c.} & Au - Av \leq b \\ & u \geq 0 \quad v \geq 0 \end{cases}$$

② => ③ $A' = (A \ \ I_m)$

$b' = b$ $x' = [x, y]$

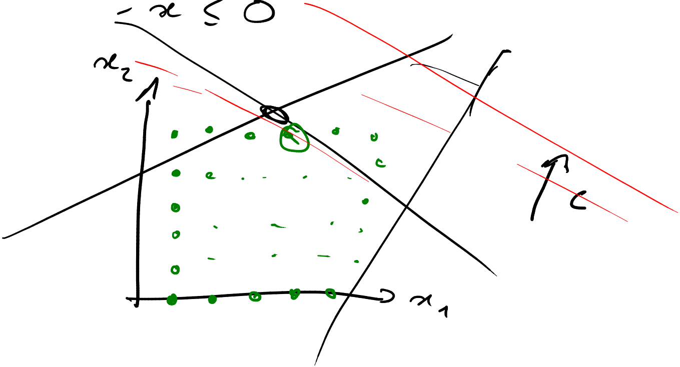
$y \in \mathbb{R}^m$ $c' = [c, 0_m]$

$$\begin{cases} \min & c^T x \\ \text{s.c.} & Ax + y = b \\ & x \geq 0 \quad y \geq 0 \end{cases}$$

$(3) \Rightarrow (1)$

$$A' = \begin{pmatrix} A \\ -A \\ -I_n \end{pmatrix} \quad b' = \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix} \quad c' = c$$

$$\begin{cases} \min_x \; c^T x \\ \text{s.c.} \quad Ax \leq b \\ \qquad -Ax \leq -b \\ \qquad -x \leq 0 \end{cases}$$



Interprétation géométrique : polyèdre

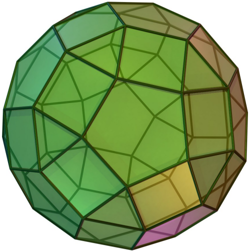


Figure – Polytope : polyèdre borné.

Théorème. PL$\in \mathcal{P}$

Base, solution basique/basique réalisable

$A = [A_B | A_N]$

Hypothèse La matrice $\mathbf{A} \in \mathbb{R}^{m \times n}$ est de rang plein (ses lignes sont linéairement indépendantes). $m \leq n$

Définition Une base $B \subset [n]$ est un sous ensemble de m indices de colonnes, tel que $\mathbf{A}_B = (a_{i,j})_{i \in [m], j \in B}$ est inversible.

Définition La **solution basique** associée à la base B est alors $\mathbf{x} = \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{pmatrix}$, où $\mathbf{x}_B = \mathbf{A}_B^{-1}\mathbf{b}$ et $\mathbf{x}_N = 0$ et $N = [n] \setminus B$.

Elle est alors **réalisable** si $\mathbf{A}_B^{-1}\mathbf{b} \geq 0$.

$$\begin{cases} \min & c^T x \\ \text{s.c.} & Ax = b \\ & x \geq 0 \end{cases}$$

$$x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} \quad c = (c_B \ c_N)$$

$$A = (A_B \ A_N)$$

$$A_B \in \mathbb{R}^{m \times m}$$

$$A_N \in \mathbb{R}^{m \times (n-m)}$$

$$x_B \in \mathbb{R}_+^m, \ x_N \in \mathbb{R}_+^{n-m}$$

$$\begin{cases} \min & c_B^T x_B + c_N^T x_N \\ \text{s.c} & A_B x_B + A_N x_N = b \\ & x \geq 0 \end{cases}$$

$$\begin{cases} \min & c_B^T x_B + c_N^T x_N \\ \text{s.c} & x_B + A_B^{-1} A_N x_N = A_B^{-1} b \\ & x \geq 0 \end{cases}$$

$$\begin{cases} \min & c_B^T A_B^{-1} b + (c_N^T - A_B^{-1} A_N) x_N \\ \text{s.c.} & x_B + A_B^{-1} A_N x_N = A_B^{-1} b \\ & x \geq 0 \end{cases}$$

Reformulation à partir d'une base

Soit B une base réalisable, $N = [n] \setminus B$, on peut écrire la forme standard :

$$
\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}_B^\top \mathbf{A}_B^{-1}\mathbf{b} + \overbrace{(\mathbf{c}_N^\top - \mathbf{c}_B^\top \mathbf{A}_B^{-1}\mathbf{A}_N)}^{\text{coûts réduits}}\mathbf{x}_N \\
s.c. \quad & \mathbf{x}_B = \mathbf{A}_B^{-1}\mathbf{b} - \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N \\
& \mathbf{x} \geq 0
\end{aligned}
$$

Les coûts réduits associés sont alors $\underline{\mathbf{r}_N} = \mathbf{c}_N^\top - \mathbf{c}_B^\top \mathbf{A}_B^{-1}\mathbf{A}_N$.

Théorèmes

Théorème. Si le PL admet une solution optimale, alors il admet une solution optimale basique réalisable.

Théorème. Soit B une base réalisable. Si $\mathbf{r}_N \geq 0$, alors B est optimale. Sinon, soit $k \in N$, tel que $r_k < 0$ on a exactement un des deux cas suivants :

1. Il existe une base réalisable $B' \subset B \cup k$ avec un objectif au moins aussi petit que B.
2. Il n'y a pas de base réalisable $B' \subset B \cup k$ différente de B, et la valeur du problème est $-\infty$.

Pivot et algorithme du simplexe

Définition. Une **règle de pivot** définit de manière non-ambigue la nouvelle base réalisable B' à partir de B.

Algorithme du simplexe

- Exhiber une base réalisable initiale $B \leftarrow B_0$.
- Tant que les coûts réduits ne sont pas positifs et qu'on est dans le cas 1 du théorème :
  Choix d'une base $B \leftarrow B'$ selon règle de pivot.
- Retourner la dernière base et la valeur du problème (potentiellement infinie).

Algorithme du simplexe

Théorème. Il existe une règle de pivot pour laquelle le simplexe ne cycle pas.

$\Longrightarrow$ nombre fini d'itérations

Remarque

L'algorithme du simplexe n'est pas un algorithme polynomial, mais est très rapide en pratique.

Illustration du simplexe

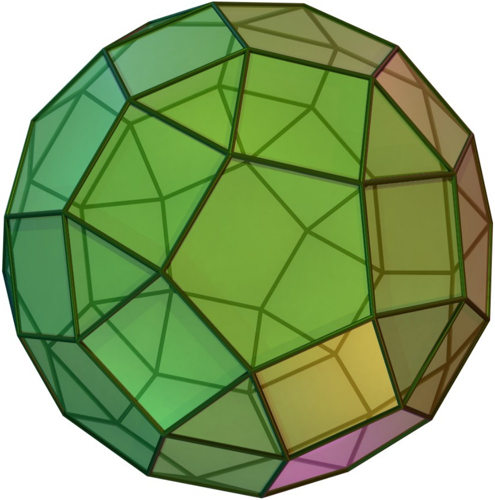


Figure – Polytope : polyèdre borné.

Problèmes primal et dual

On rappelle la forme standard du primal :

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}^\top \mathbf{x} \\ s.c. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \quad (y \in \mathbb{R}^m) \\ & \mathbf{x} \geq 0 \end{aligned} \tag{P}$$

Son dual est également un programme linéaire :

$$\begin{aligned} \max_{\mathbf{y}\in\mathbb{R}^m} \quad & \mathbf{b}^\top \mathbf{y} \\ s.c. \quad & \mathbf{A}^\top \mathbf{y} \leq \mathbf{c} \quad (x \geqslant 0) \end{aligned} \tag{D}$$

Théorème. (D) est le dual lagrangien de (P), et inversement → exercice.

$$\mathcal{L}(x,y) = c^T x + y^T (b - Ax)$$

$$\mathrm{val}(P) = \min_x \max_y \mathcal{L}(x,y)$$

$$\mathrm{val}(D) = \max_y \min_x \mathcal{L}(x,y)$$

$$\mathcal{L}(x,y) = \underbrace{b^T y} + \underbrace{x^T}_{x \geq 0} \underbrace{\left(c - A^T y\right)}_{\geq 0}$$

$$\begin{cases} \max\limits_y \; b^T y \\ \text{s.c.} \quad A^T y \leq c \\ \qquad\quad y \in \mathbb{R}^m \end{cases}$$

Dualité forte

Dualité faible :
$\mathrm{val}(D) \leq \mathrm{val}(P)$

Théorème. Une et une seule des assertions suivantes est vraie pour (P) et (D) :

1. Ni (P) ni (D) n'a de solution réalisable.
2. (P) est non borné et (D) n'a pas de solution réalisable.
3. (D) est non borné et (P) n'a pas de solution réalisable.
4. (P) et (D) ont une solution réalisable, et dans ce cas $\boxed{\mathrm{val}(P) = \mathrm{val}(D).}$

- Dualité faible :

$$\min_x \mathcal{L}(x,y') \leq \max_y \mathcal{L}(x',y) \quad \forall x', y'$$

$$\max_y \min_x \mathcal{L}(x,y) \leq \max_y \mathcal{L}(x',y) \quad \forall x'$$

$$\underbrace{\max_y \min_x \mathcal{L}(x,y)}_{val(D)} \leq \underbrace{\min_x \max_y \mathcal{L}(x,y)}_{val(P)}$$

- Cas linéaire : soit B une base optimale pour P

$$x^* = \begin{pmatrix} x_B \\ x_N \end{pmatrix}, \quad \begin{array}{l} x_B = A_B^{-1} b \\ x_N = 0 \end{array}, \quad y = (A_B^{-1})^T c_B$$

$$A^T y = \begin{pmatrix} A_B^T \\ A_N^T \end{pmatrix} y = \begin{pmatrix} A_B^T (A_B^{-1})^T c_B = c_B \\ A_N^T (A_B^{-1})^T c_B \end{pmatrix} \leq \begin{pmatrix} c_B^T \\ c_N^T \end{pmatrix}$$

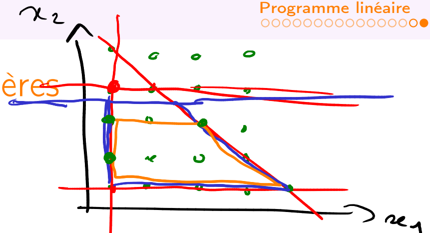
car $\pi_N = c_N^T - c_B^T A_B^{-1} A_N \geqslant 0 \rightarrow y$ réalisable pour D

$$val(P) = c_B A_B^{-1} b = b^T y = b^T (A_B^{-1})^T c_B$$

Donc y optimal et $val\, P = val\, D$

Preuve du point 4.

Polyèdre entier et solutions entières



P entier

P' non entier

$\text{conv}(P' \cap \mathbb{Z}^n)$

Définition Un polyèdre P est **entier** si $P = \text{conv}(P \cap \mathbb{Z}^n)$.

Théorème. Soit P un polyèdre, on a l'équivalence entre :

1. P est entier.
2. $\min\{\mathbf{c}^\top \mathbf{x}, \mathbf{x} \in P\}$ est atteint en un vecteur $\mathbf{x}^* \in \mathbb{Z}^n$ pour tout vecteur de coût $\mathbf{c}$ pour lequel le programme linéaire est fini.
3. $\min\{\mathbf{c}^\top \mathbf{x}, \mathbf{x} \in P\}$ est une valeur entière pour tout $\mathbf{c} \in \mathbb{Z}^n$ tel que le minimum est fini.

Totale unimodularité

Définition Une matrice $\mathbf{A} \in \mathbb{Z}^{m \times n}$ est **totalement unimodulaire** si le déterminant de toutes ses sous matrices carrées est dans $\{-1, 0, 1\}$.

Théorème. Une matrice $\mathbf{A} \in \mathbb{Z}^{m \times n}$ est totalement unimodulaire si et seulement si le polyèdre $P = \{\mathbf{x} \in \mathbb{R}^n, \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq 0\}$ est entier ~~intégral~~ pour tout $\mathbf{b} \in \mathbb{Z}^m$.

Théorème. Soit $\mathbf{A}$ une matrice avec coefficients dans $\{-1, 0, 1\}$. Si elle contient au plus un 1 et un -1 par colonne, alors elle est totalement unimodulaire. $\longrightarrow$ flots