Contourner l'explosion combinatoire

Pierre Fouilhoux

Université Pierre et Marie Curie
Laboratoire LIP6 - Equipe Recherche Opérationnelle

Séminaire de vulgarisation Normandie-Mathématiques
27 janvier 2016

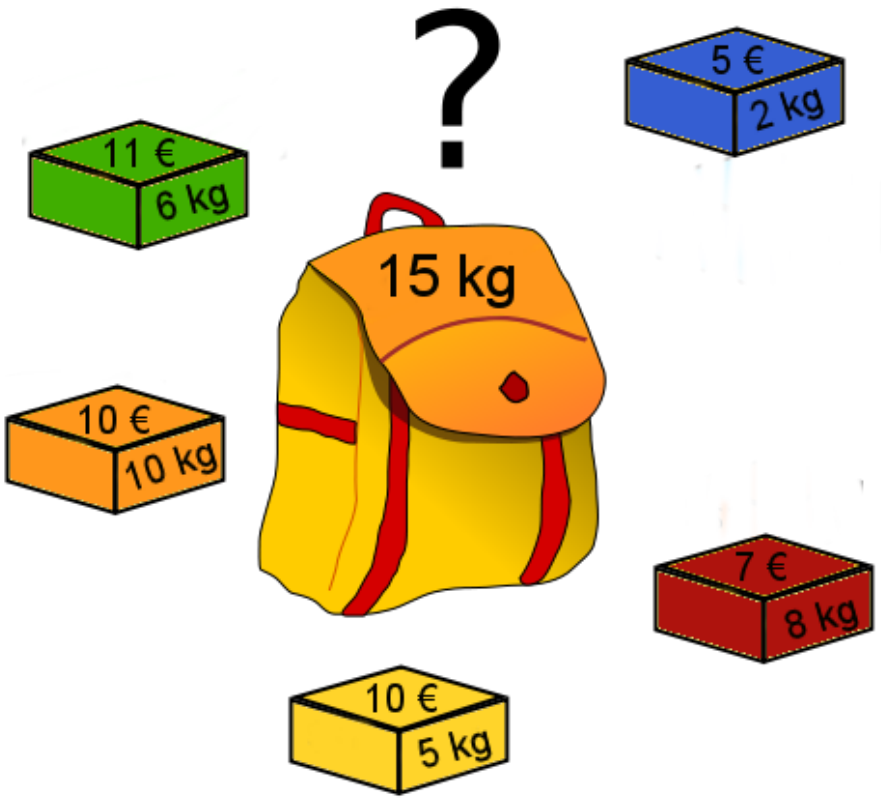
Un **problème d'optimisation combinatoire**

consiste à

trouver un plus petit (ou plus grand) élément
dans un ensemble fini valué.

Le problème du sac-à-dos

Quelles boîtes choisir
en maximisant le gain
sans dépasser 15kg ?



Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

On veut trouver un sous-ensemble $S \subset \{1, \ldots, n\}$ tel que

Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

On veut trouver un sous-ensemble $S \subset \{1, \ldots, n\}$ tel que

son poids $\sum_{i \in S} p_i$ soit inférieur à P *(Contrainte de sac-à-dos)*

Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

On veut trouver un sous-ensemble $S \subset \{1, \ldots, n\}$ tel que

son poids $\sum_{i \in S} p_i$ soit inférieur à P *(Contrainte de sac-à-dos)*

son gain $\sum_{i \in S} g_i$ soit maximum. *(Fonction "objective")*

Comment représenter une solution du problème ?

Instance donnée par :

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Une solution $S \subset \{1, \ldots, n\}$ correspond à un **vecteur d'incidence** χ^S tel que $\chi^S[i] = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$

$S_1 = \{1, 2, 5\}$

$\chi^{S_1} =$

1	1	0	0	1

Comment représenter une solution du problème ?

Instance donnée par :

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Une solution $S \subset \{1, \ldots, n\}$ correspond à un **vecteur d'incidence** χ^S

tel que $\chi^S[i] = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$

$S_1 = \{1, 2, 5\}$

$\chi^{S_1} =$

1	1	0	0	1

coût :	5	7	10	$= 22$
poids :	2	8	5	$= 15$

Comment reconnaître si un vecteur est solution ?

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_2 = \{1, 4, 5\}$$

$$\chi^{S_2} = \begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 1 & 1 \\ \hline \end{array}$$

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_2 = \{1, 4, 5\}$$

$$\chi^{S_2} = \boxed{1}\ \boxed{0}\ \boxed{0}\ \boxed{1}\ \boxed{1}$$

$$\text{poids :} \quad 2 \qquad\qquad 6 \quad 5 \quad = 13$$

Solution de coût 26

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_3 = \{1, 2, 4\}$$

$$\chi^{S_3} = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_3 = \{1, 2, 4\}$$

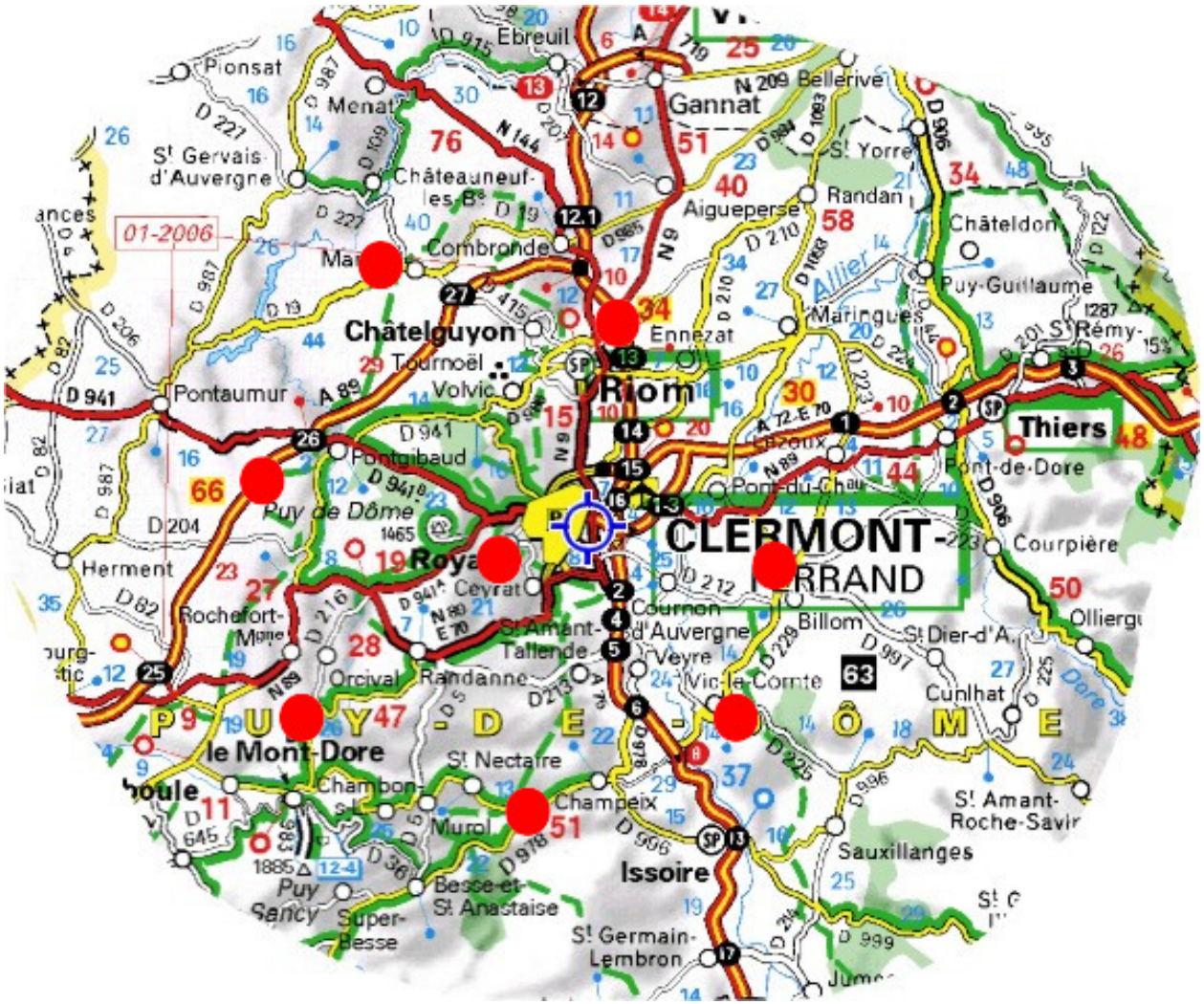
$$\chi^{S_3} = \boxed{1 \mid 1 \mid 0 \mid 1 \mid 0}$$

$$\text{poids} : \quad 2 \quad 8 \qquad\qquad 6 \qquad\qquad = 16$$

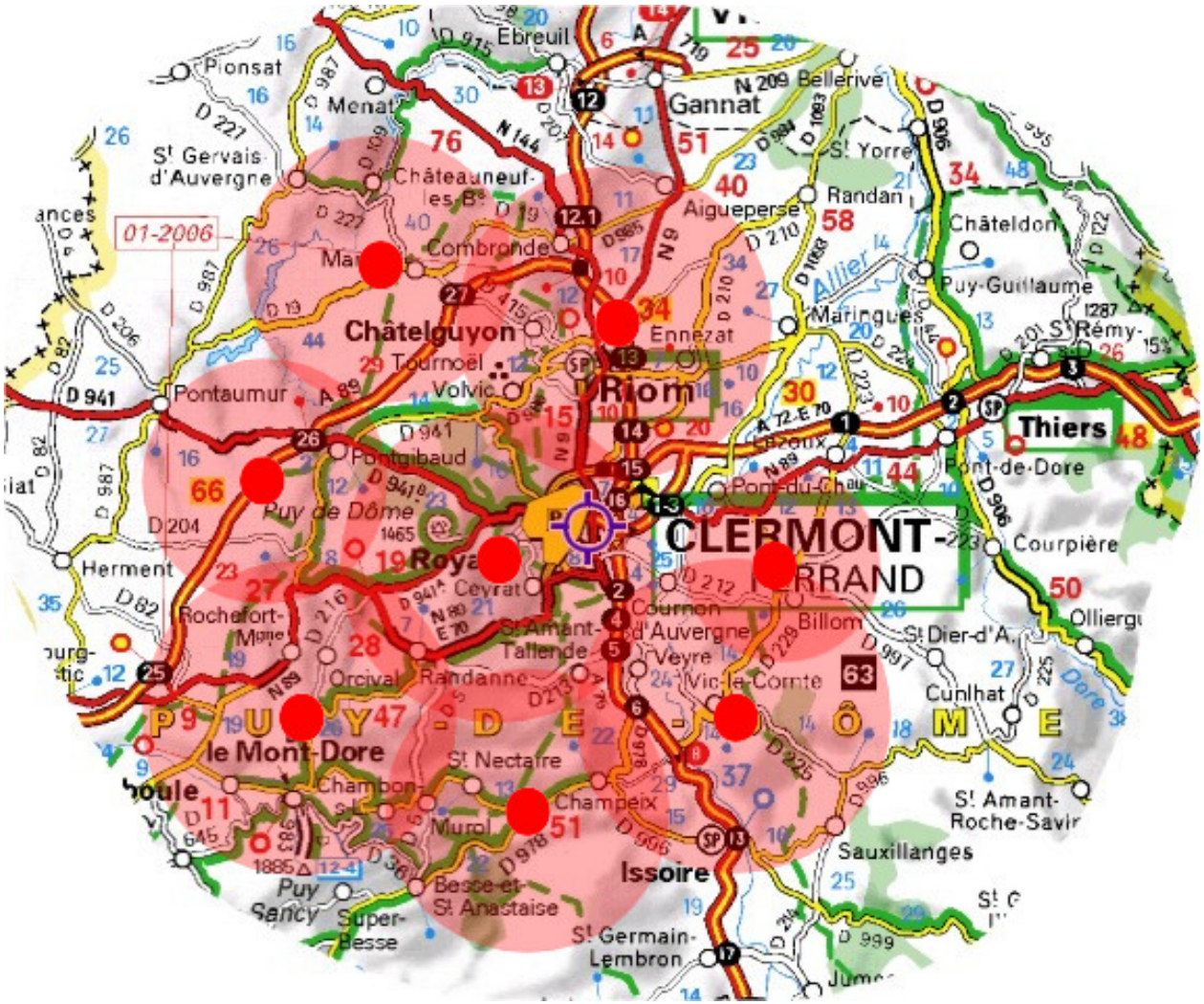
Pas solution

De grands sac-à-dos...

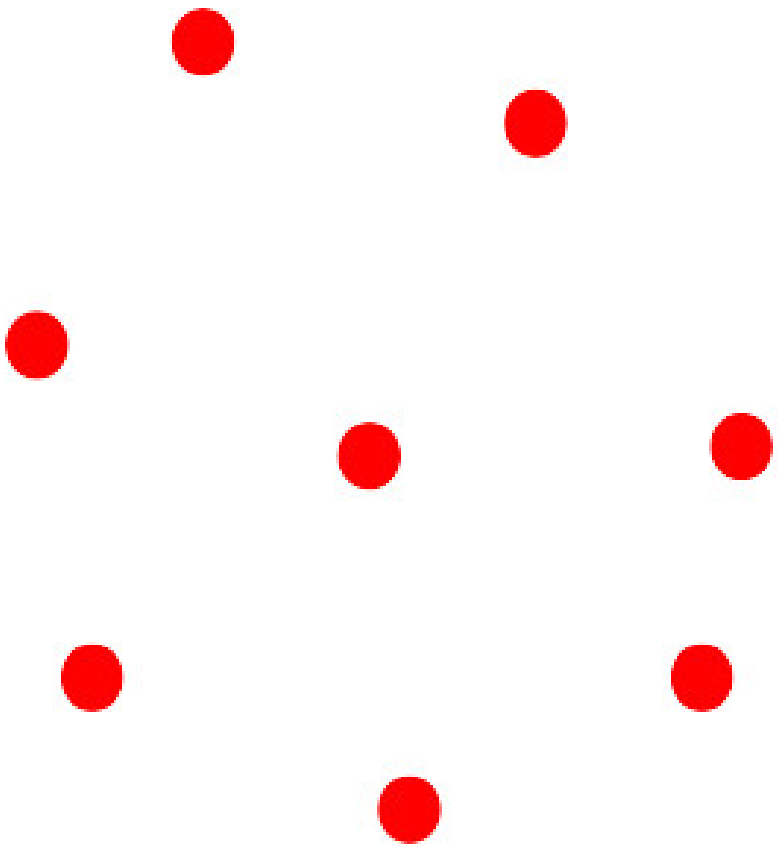
Couverture hertzienne



Couverture hertzienne

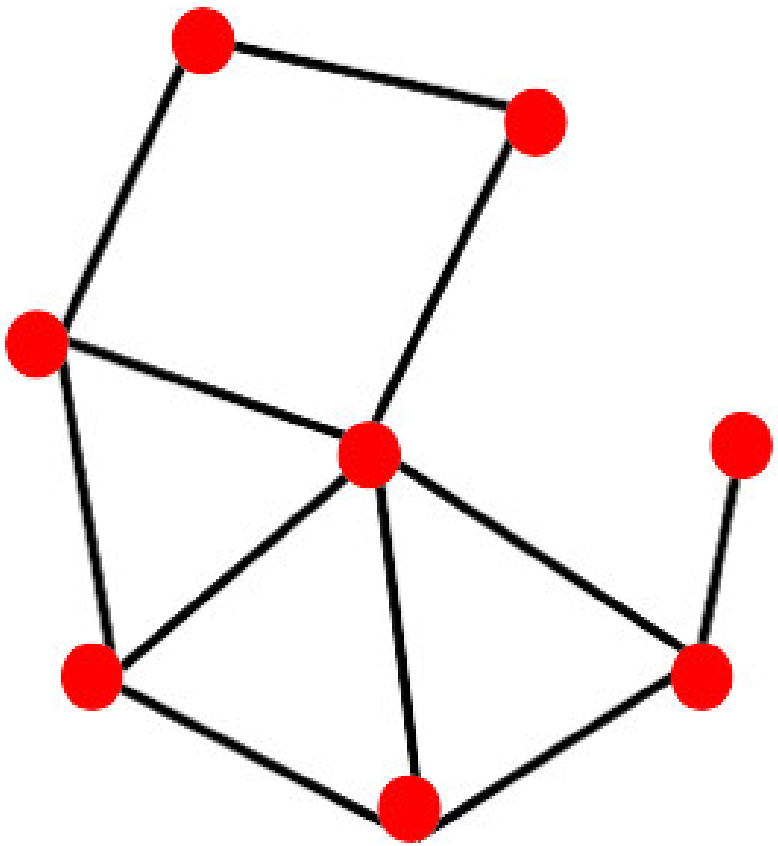


Modélisation par un graphe



Un graphe $G = (S, A)$
avec
S : ensemble des sommets

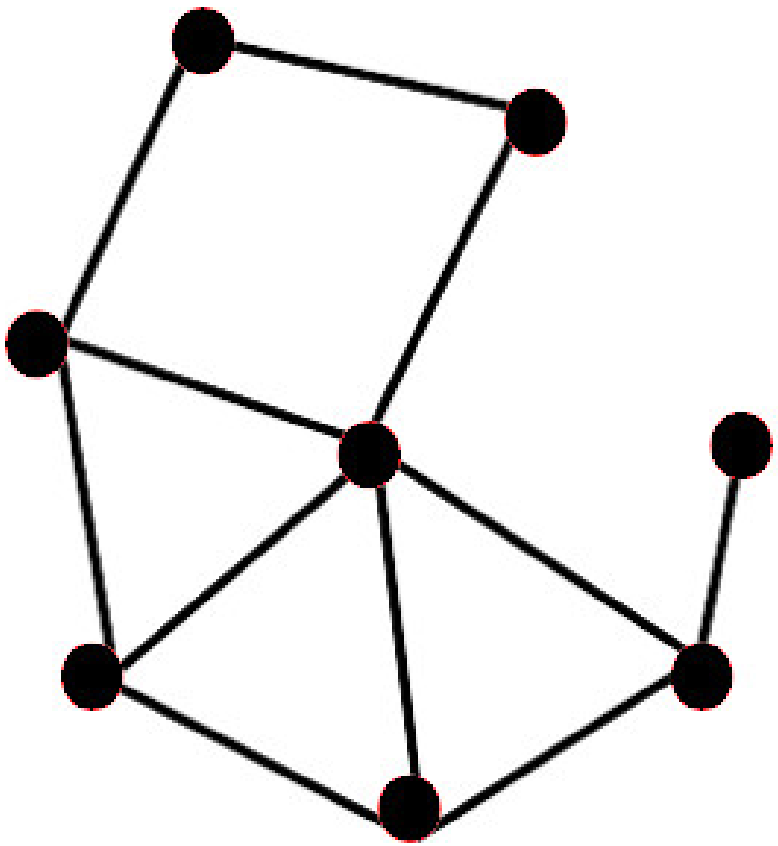
Modélisation par un graphe



Un graphe $G = (S, A)$
avec
S : ensemble des sommets
A : ensemble des arêtes

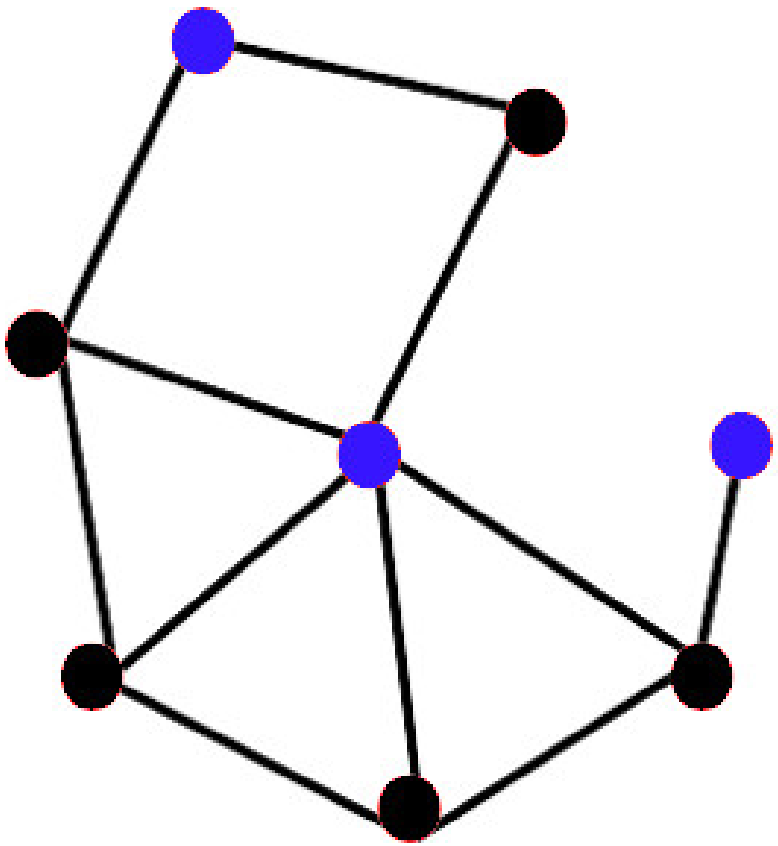
Une arête représente que le conflit entre deux arêtes trop proches : on ne peut leur donner la même fréquence.

Problème du stable



Un **stable** est un ensemble de sommet non reliés par des arêtes.

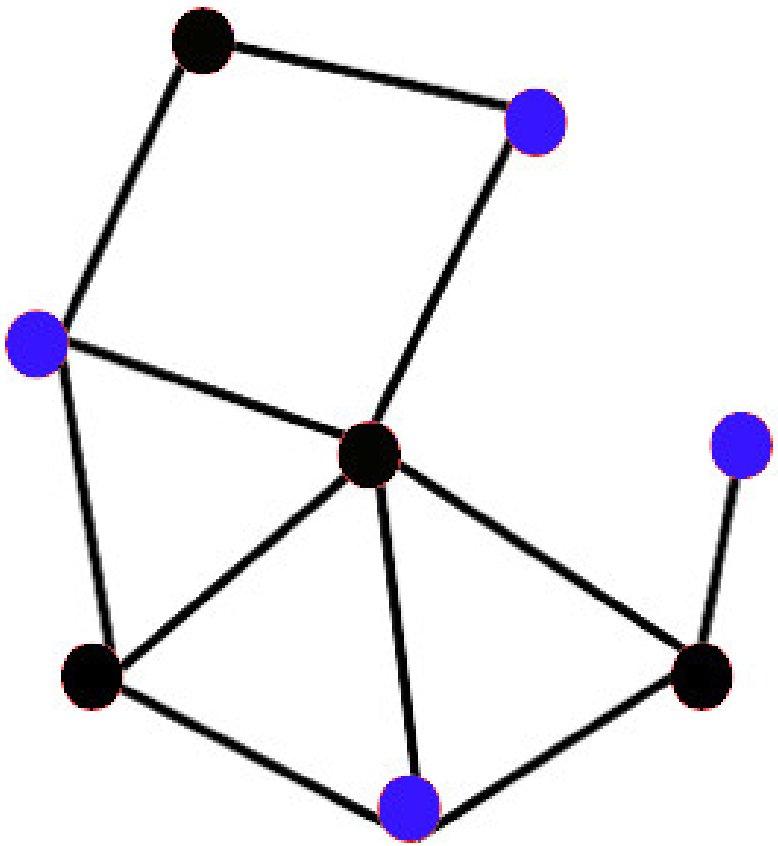
Problème du stable



Un **stable** est un ensemble de sommet non reliés par des arêtes.

Le **Problème du stable maximum** consiste à déterminer le stable ayant le plus de sommets possible.

Problème du stable

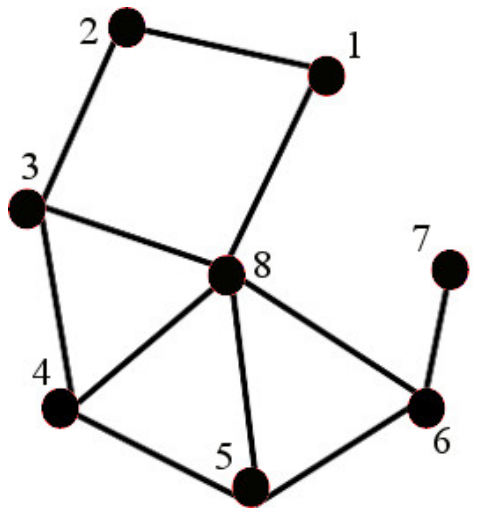


Un **stable** est un ensemble de sommet non reliés par des arêtes.

Le **Problème du stable maximum** consiste à déterminer le stable ayant le plus de sommets possible.

Comment représenter une solution du problème ?

Instance donnée par
un graphe avec
n sommets



Une solution $S \subset \{1, \ldots, n\}$
correspond à un
vecteur d'incidence χ^S
tel que $\chi^S[i] = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$

$S_1 = \{1, 7, 8\}$

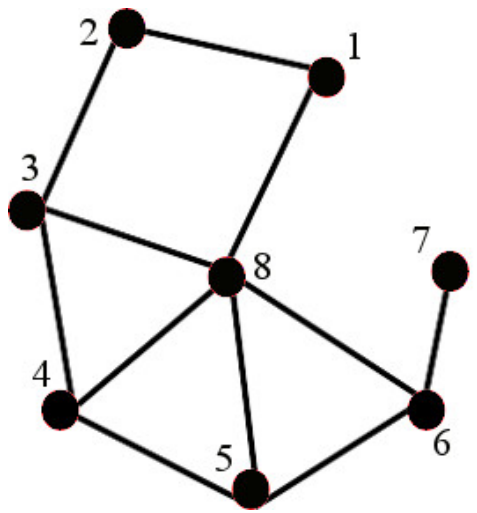
$\chi^{S_1} =$

i	1	2	3	4	5	6	7	8
	1	0	0	0	0	0	1	1

Comment reconnaître si un vecteur est solution ?

Algorithme de reconnaissance :

Pour i de 1 à n
  Pour j de 1 à n
    Si $\{i, j\}$ est une arête
      et si $\chi^S[i] = 1$ et $\chi^S[j] = 1$
    Alors STOP : Faux
  FinPour
FinPour
STOP : Vrai



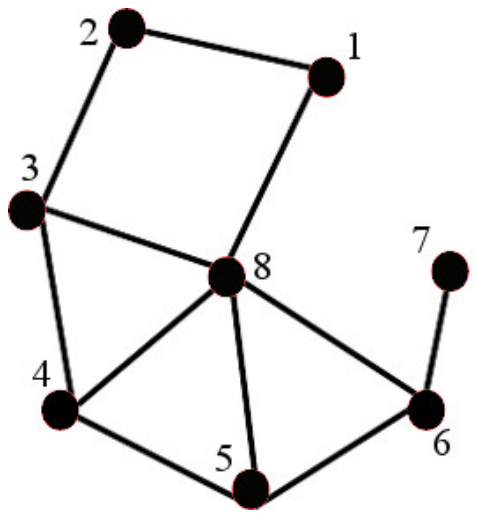
$S_1 = \{2, 7, 8\}$

i	1	2	3	4	5	6	7	8
$\chi^{S_1} =$	0	1	0	0	0	0	1	1

Comment reconnaître si un vecteur est solution ?

Algorithme de reconnaissance :

Pour i de 1 à n
  Pour j de 1 à n
    Si $\{i, j\}$ est une arête
      et si $\chi^S[i] = 1$ et $\chi^S[j] = 1$
    Alors STOP : Faux
  FinPour
FinPour
STOP : Vrai



$S_1 = \{2, 7, 8\}$

i	1	2	3	4	5	6	7	8
$\chi^{S_1} =$	0	1	0	0	0	0	1	1

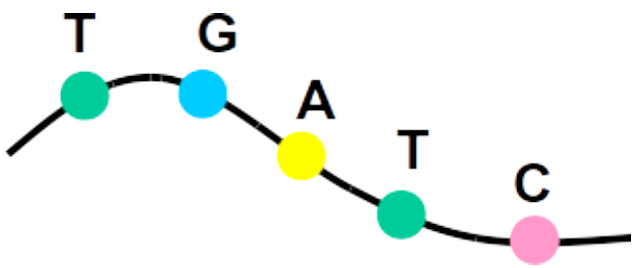
Solution (avec 3 sommets)

Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

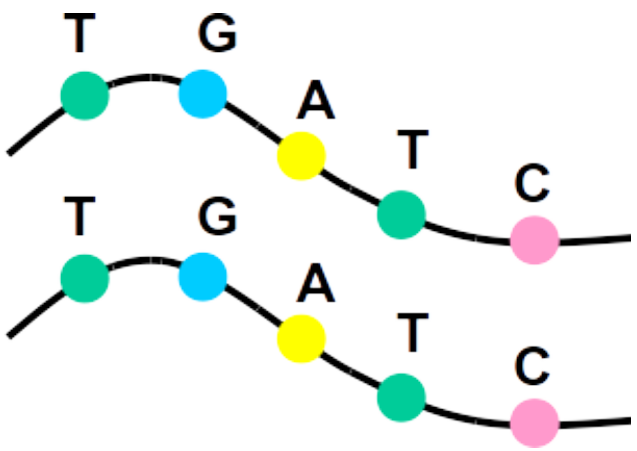


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

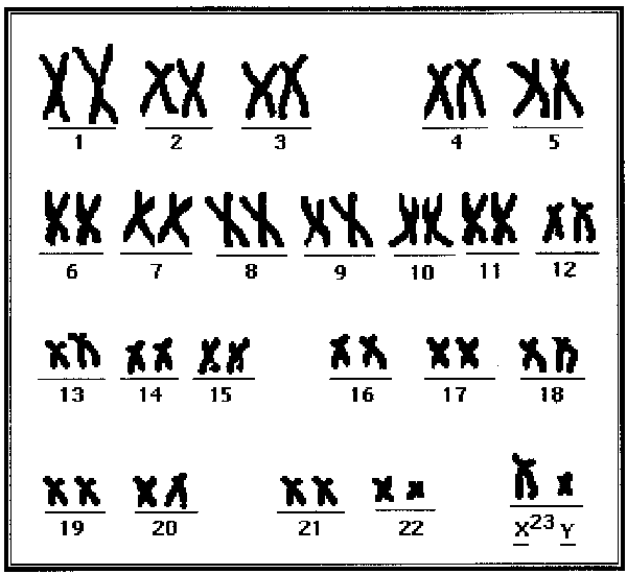


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

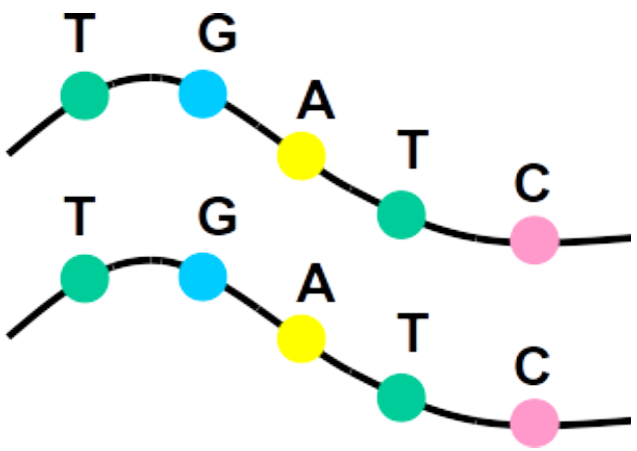


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

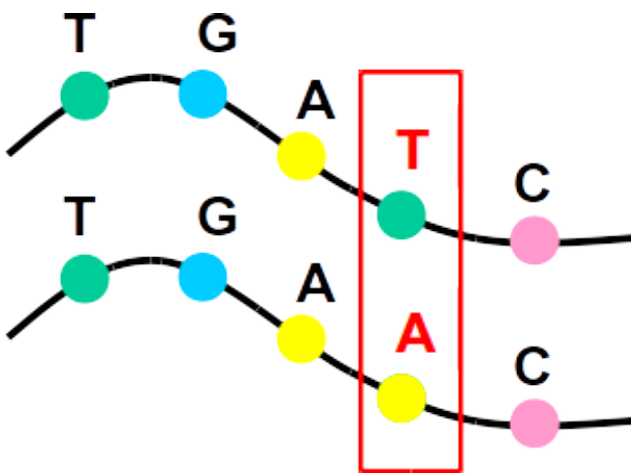


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

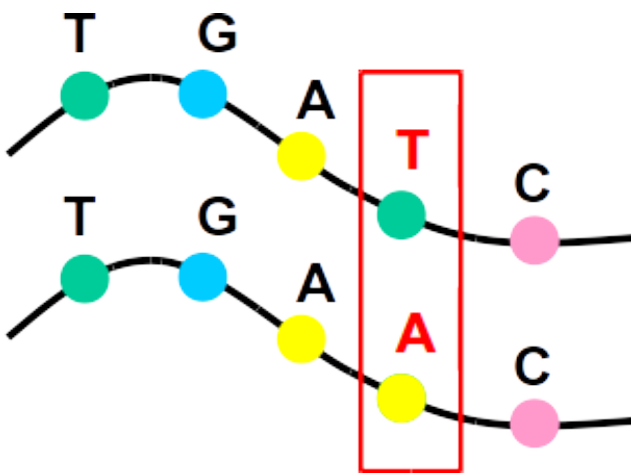


Combinatoire du génôme

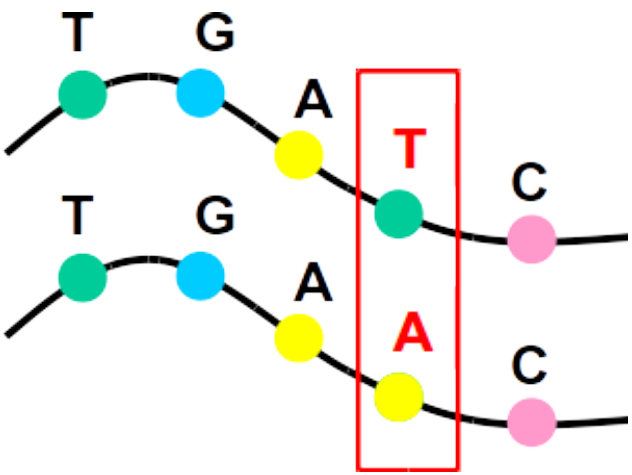
L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).



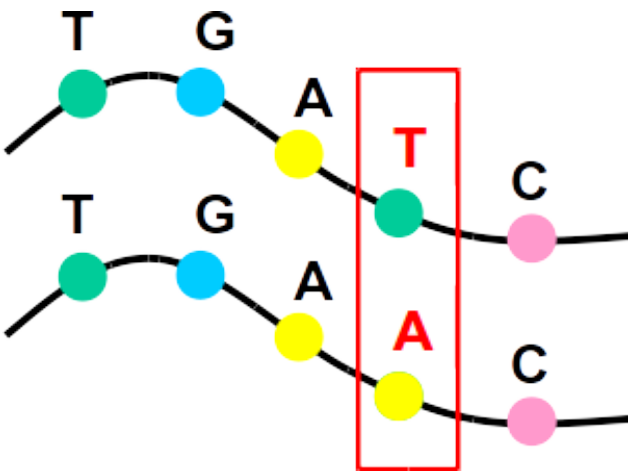
Combinatoire du génome



Lors du séquençage du génome, on doit trier parmi des petis fragments d'ADN qui ont été séquencé par des procédés bio-mécaniques.
Lors de l'opération, des SNPs faux ont été créés

On veut trier les fragments en omettant le moint de SNPs possible.

Combinatoire du génome

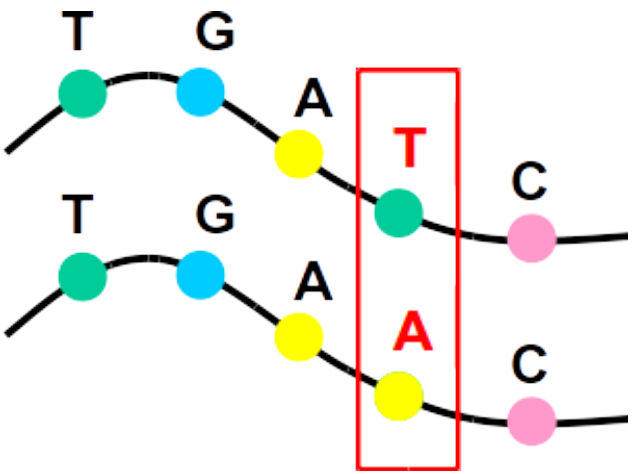


Lors du séquençage du génome, on doit trier parmi des petis fragments d'ADN qui ont été séquencé par des procédés bio-mécaniques.
Lors de l'opération, des SNPs faux ont été créés

On veut trier les fragments en omettant le moint de SNPs possible.

C'est un problème **d'optimisation combinatoire !**

Combinatoire du génome



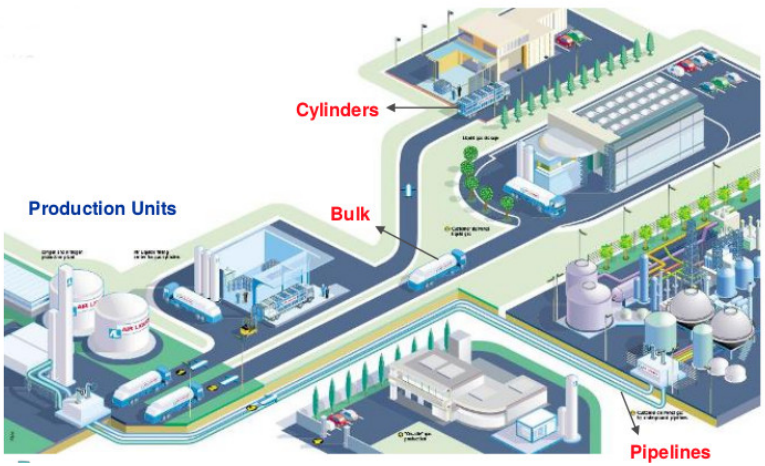
Lors du séquençage du génome, on doit trier parmi des petis fragments d'ADN qui ont été séquencé par des procédés bio-mécaniques.
Lors de l'opération, des SNPs faux ont été créés

On veut trier les fragments en omettant le moint de SNPs possible.

C'est un problème **d'optimisation combinatoire !**
Il a été montré que ce problème est exactement celui du **stable maximum**.

Lippert, R., Schwartz, R., Lancia, G., Istrail, S. : Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Brief. Bioinform 3, 23–31 (2002)

Problème posé pour le Challenge Roadef 2016



L'entreprise **Air Liquide** propose ce problème réel :
Déterminer les tournées des camions d'oxygène liquide pour délivrer les hôpitaux
de manière à ce que :
- les cuves des hopitaux ne soient jamais vide ("monitoring" à distance)
- les tournées soient faisables dans la journée de travail du conducteur
- les coûts soient minimisés !

Problème d'optimisation combinatoire :

Etant donnés :
un ensemble fini d'éléments $\{1, ..., n\}$
avec un coût par élément $c_1, ..., c_n$

Problème d'optimisation combinatoire :

Etant donnés :
un ensemble fini d'éléments $\{1, ..., n\}$
avec un coût par élément $c_1, ..., c_n$

Et une famille $\mathcal{F}$ de sous-ensembles de $\{1, \ldots, n\}$
Ces sous-ensemble sont appelés **solutions**

Trouver un ensemble $F \in \mathcal{F}$ de poids $\sum_{i \in F} c_i$ maximum (ou minimum)

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance Stable

Pour i de 1 à n

Pour j de 1 à n

Si $\{i, j\}$ est une arête

et si $\chi^S[i] = \chi^S[j] = 1$

Alors STOP : Faux

FinPour

FinPour

STOP : Vrai

Son temps d'exécution est proportionnel à n^2.

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance Stable

Pour i de 1 à n

Pour j de 1 à n

Si $\{i, j\}$ est une arête

et si $\chi^S[i] = \chi^S[j] = 1$

Alors STOP : Faux

FinPour

FinPour

STOP : Vrai

Son temps d'exécution est proportionnel à n^2.

Les algorithmes **rapides** ont un temps d'exécution proportionnel à n ou à n^2 ou à n^3 ou à n^4,..., ou à n^{12},...
que l'on appelle **Algorithmes Polynomiaux**.

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance Stable

Pour i de 1 à n

Pour j de 1 à n

Si $\{i, j\}$ est une arête

et si $\chi^S[i] = \chi^S[j] = 1$

Alors STOP : Faux

FinPour

FinPour

STOP : Vrai

Son temps d'exécution est proportionnel à n^2.

Les algorithmes **très très lents** ont un temps d'exécution proportionnel à 2^n, 3^n,... que l'on appelle **Algorithmes exponentiels**.

Les algorithmes **rapides** ont un temps d'exécution proportionnel à n ou à n^2 ou à n^3 ou à n^4,..., ou à n^{12},... que l'on appelle **Algorithmes Polynomiaux**.

Enumérer ?

Prenons un problème d'optimisation avec n éléments.
Supposons que l'on connaisse un algorithme **polynomial** (donc rapide) pour reconnaître une solution

Peut-on déterminer la meilleure solution par **énumération** ?

Enumérer ?

Prenons un problème d'optimisation avec n éléments.
Supposons que l'on connaisse un algorithme **polynomial** (donc rapide) pour reconnaître une solution

Peut-on déterminer la meilleure solution par **énumération** ?

Pour chaque sous-ensemble S d'éléments
  $\chi^S \leftarrow$ le vecteur d'incidence de S.
  Algorithme de réconnaissance pour χ^S.
  Si χ^S est solution
    Conserver chi^S si meilleure solution rencontrée.
FinPour
Retourner la meilleure solution rencontrée.

Enumérer ?

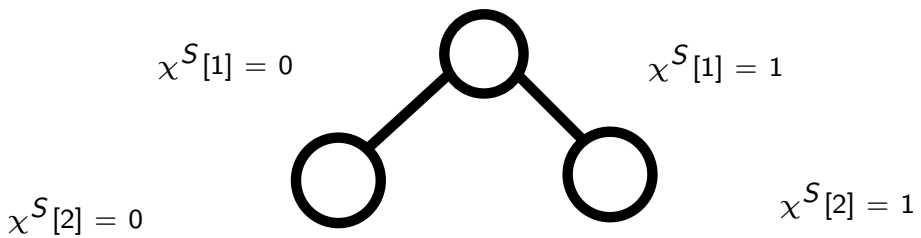
Prenons un problème d'optimisation avec n éléments.
Supposons que l'on connaisse un algorithme **polynomial** (donc rapide) pour reconnaître une solution

Peut-on déterminer la meilleure solution par **énumération** ?

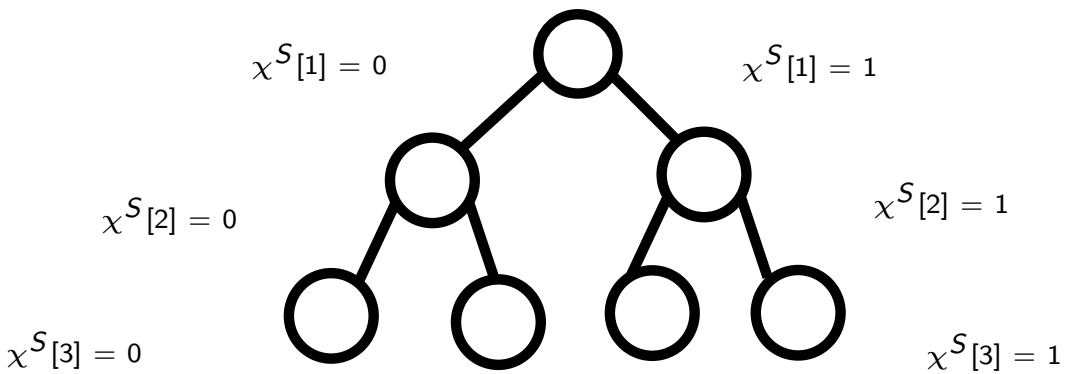
Pour chaque sous-ensemble S d'éléments
  $\chi^S \leftarrow$ le vecteur d'incidence de S.
  Algorithme de réconnaissance pour χ^S.
  Si χ^S est solution
    Conserver chi^S si meilleure solution rencontrée.
FinPour
Retourner la meilleure solution rencontrée.

Il y a 2^n **sous-ensembles** : Temps d'exécution **exponentiel**

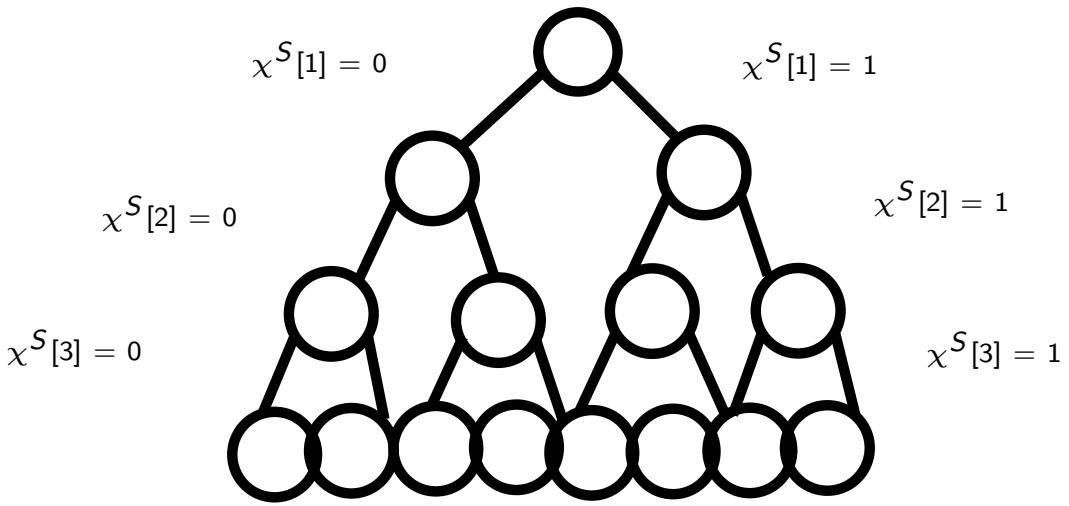
Explosion combinatoire



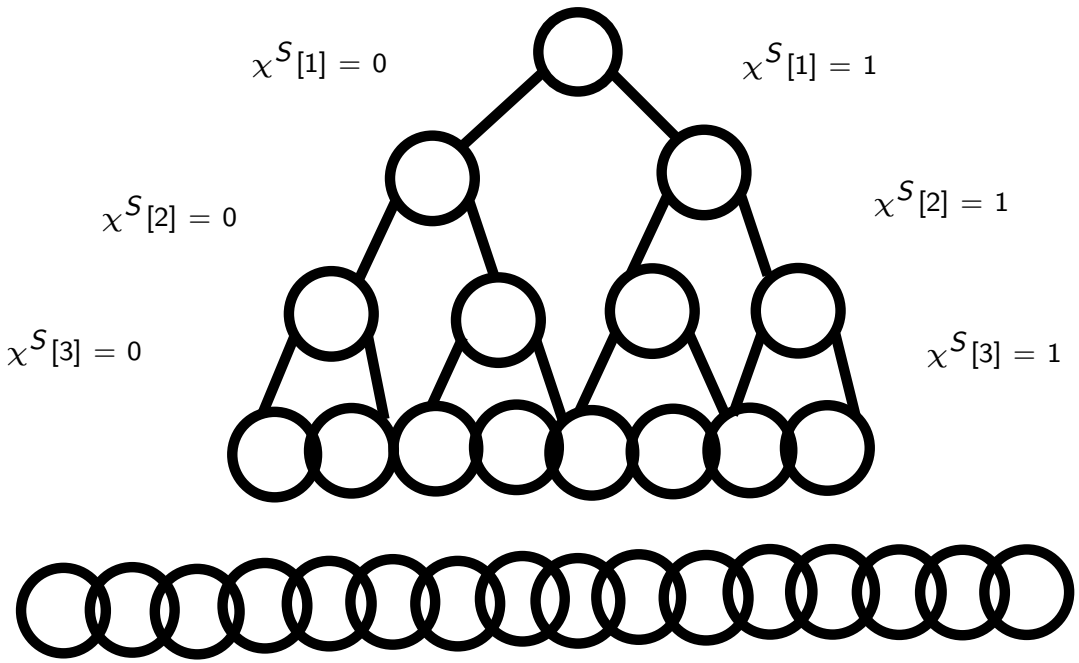
Explosion combinatoire



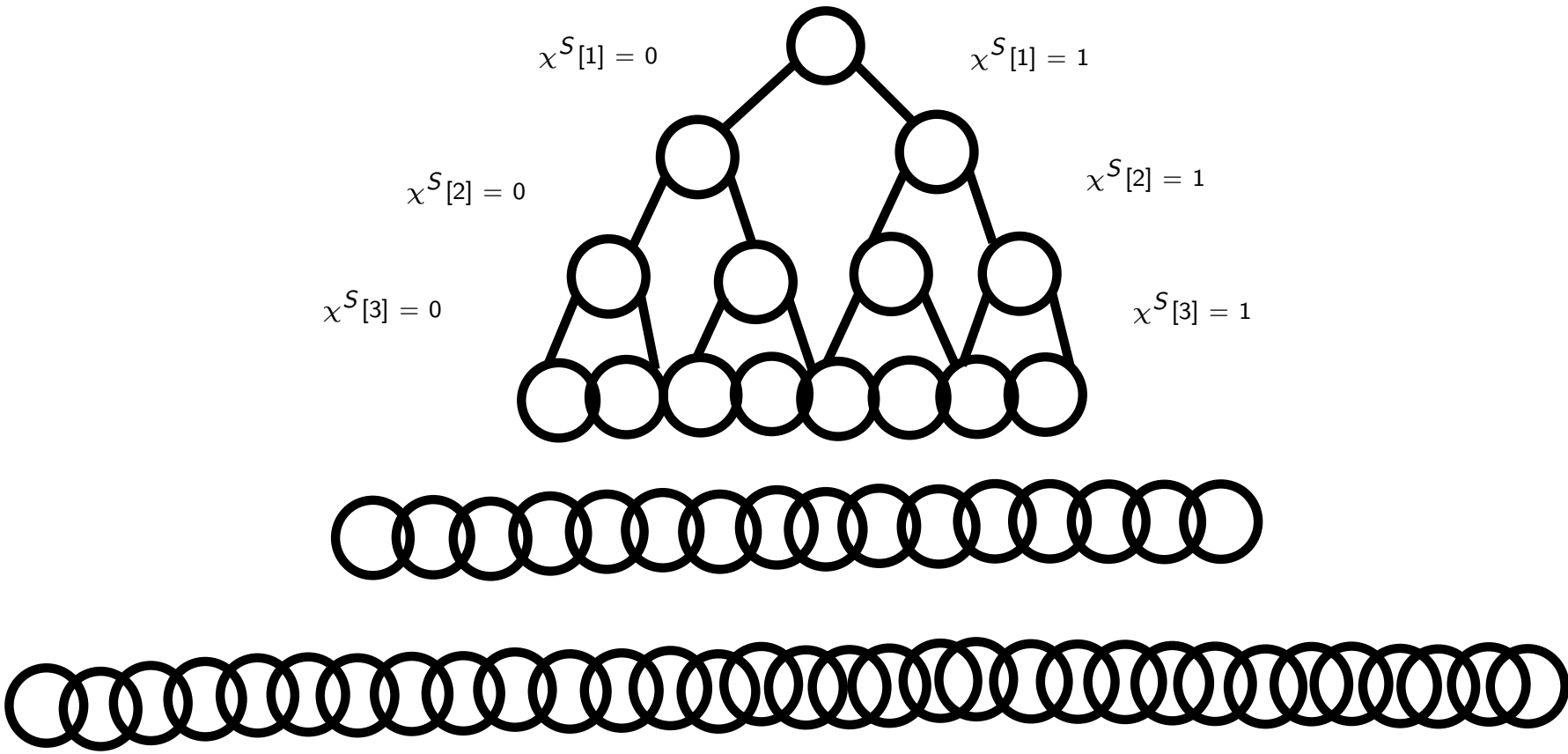
Explosion combinatoire



Explosion combinatoire



Explosion combinatoire



Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec		
20	0,00...001 sec	0,00...001 sec		
50	0,00...001 sec	0,00...001 sec		
60	0,00...001 sec	0,00000002 sec		
80	0,00...001 sec	0,0000009 sec		
100	0,00...001 sec	0,0000003 sec		
1000	0,00000003 sec	0,03 sec		

Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec	0,00...001 sec	
20	0,00...001 sec	0,00...001 sec	0,00000006 sec	
50	0,00...001 sec	0,00...001 sec	168,9 sec	
60	0,00...001 sec	0,00000002 sec	57,6 h	
80	0,00...001 sec	0,0000009 sec	9200 ans	
100	0,00...001 sec	0,0000003 sec		
1000	0,00000003 sec	0,03 sec		

Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec	0,00...001 sec	
20	0,00...001 sec	0,00...001 sec	0,00000006 sec	
50	0,00...001 sec	0,00...001 sec	168,9 sec	
60	0,00...001 sec	0,00000002 sec	57,6 h	
80	0,00...001 sec	0,0000009 sec	9200 ans	
100	0,00...001 sec	0,0000003 sec	$1,21.10^{10}$ ans	
1000	0,00000003 sec	0,03 sec	1.10^{282} ans	

Age de l'univers $13,7.10^9$ ans

Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec	0,00...001 sec	0,00...001 sec
20	0,00...001 sec	0,00...001 sec	0,00000006 sec	0,00...001 sec
50	0,00...001 sec	0,00...001 sec	168,9 sec	0,000001 sec
60	0,00...001 sec	0,00000002 sec	57,6 h	0,0001 sec
80	0,00...001 sec	0,0000009 sec	9200 ans	100 ans
100	0,00...001 sec	0,0000003 sec	$1,21.10^{10}$ ans	$1,21.10^{9}$ ans
1000	0,00000003 sec	0,03 sec	1.10^{282} ans	...

Age de l'univers $13,7.10^{9}$ ans

Explosion combinatoire

Enumérer 2^n solutions n'est donc pas un problème technique que des ordinateurs très puissants pourraient balayer.

Il est nécessaire de **contourner l'explosion combinatoire** par des outils mathématiques et algorithmiques.

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

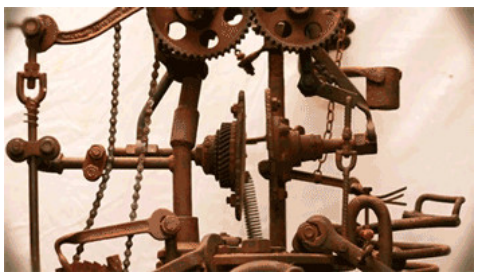
Pour casser une noix, on peut :

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :



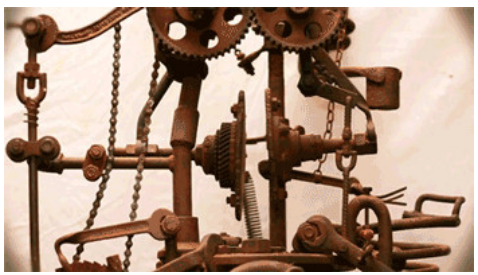


Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :



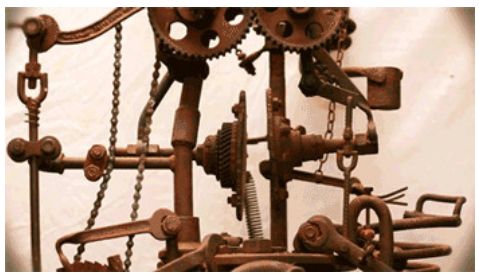



Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :






On cherche l'algorithme **le plus rapide** pour résoudre un problème !

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Un problème est **de complexité polynomiale**
s'il existe un algorithme **polynomial** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{P}$.

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Un problème est **de complexité polynomiale**
s'il existe un algorithme **polynomial** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{P}$.

Donc $\mathcal{P}$ est **incluse** dans $\mathcal{EXP}$, on note $\mathcal{P} \subset \mathcal{EXP}$

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Un problème est **de complexité polynomiale**
s'il existe un algorithme **polynomial** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{P}$.

Donc $\mathcal{P}$ est **incluse** dans $\mathcal{EXP}$, on note $\mathcal{P} \subset \mathcal{EXP}$

Question : Dans quelles classes de complexité sont les problèmes d'optimisation combinatoire ?

Problème $\mathcal{NP}$

On crée un classe un peu particulière...

Les problèmes d'optimisation combinatoire pour lesquels
on sait **reconnaître** par un algorithme polynomial
si un sous-ensemble d'éléments est solution

$\Rightarrow$ la classe de problème $\mathcal{NP}$ "Nondeterministic polynomial time"

Problème $\mathcal{NP}$

On crée un classe un peu particulière...

Les problèmes d'optimisation combinatoire pour lesquels
on sait **reconnaître** par un algorithme polynomial
si un sous-ensemble d'éléments est solution

$\Rightarrow$ la classe de problème $\mathcal{NP}$ "Nondeterministic polynomial time"

Sous cette hypothèse, on a vu qu'il **possible** d'utiliser un
algorithme d'énumération exponentiel

$\Rightarrow \mathcal{NP} \subset \mathcal{EXP}$

Problème $\mathcal{NP}$

On crée un classe un peu particulière...

Les problèmes d'optimisation combinatoire pour lesquels on sait **reconnaître** par un algorithme polynomial si un sous-ensemble d'éléments est solution

$\Rightarrow$ la classe de problème $\mathcal{NP}$ "Nondeterministic polynomial time"

Sous cette hypothèse, on a vu qu'il **possible** d'utiliser un algorithme d'énumération exponentiel

$\Rightarrow \mathcal{NP} \subset \mathcal{EXP}$

De plus, un problème polynomial est directement $\mathcal{NP}$

$\Rightarrow \mathcal{P} \subset \mathcal{NP} \subset \mathcal{EXP}$

Et alors ?

$\mathcal{P}$ est-il égal à $\mathcal{NP}$?

Que l'on peut traduire par :

"Existe-t-il un algorithme polynomial pour résoudre les problèmes d'optimisation combinatoire ?"

Réponse :

Et alors ?

$\mathcal{P}$ est-il égal à $\mathcal{NP}$?

Que l'on peut traduire par :

"Existe-t-il un algorithme polynomial pour résoudre les problèmes d'optimisation combinatoire ?"

Réponse : On n'en sait rien !

A l'échelle de l'humanité, on ne connaît que cet algorithme d'énumération !

C'est un des 7 problèmes du "Millenium Prize Problems" du Clay Mathematics Institute of Cambridge, Massachussetts, doté d'un million de dollar !

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

La solution est :
- trier les n objets du plus chers au moins chers
- de prendre les objets un à un dans cet ordre tant que cela rentre dans le sac !

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

La solution est :
- trier les n objets du plus chers au moins chers
- de prendre les objets un à un dans cet ordre tant que cela rentre dans le sac !

Cet algorithme **glouton** est polynomial (de l'ordre de n^2).

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

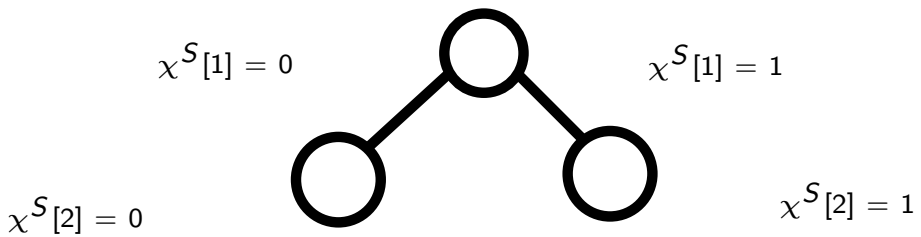
Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

La solution est :
- trier les n objets du plus chers au moins chers
- de prendre les objets un à un dans cet ordre tant que cela rentre dans le sac !

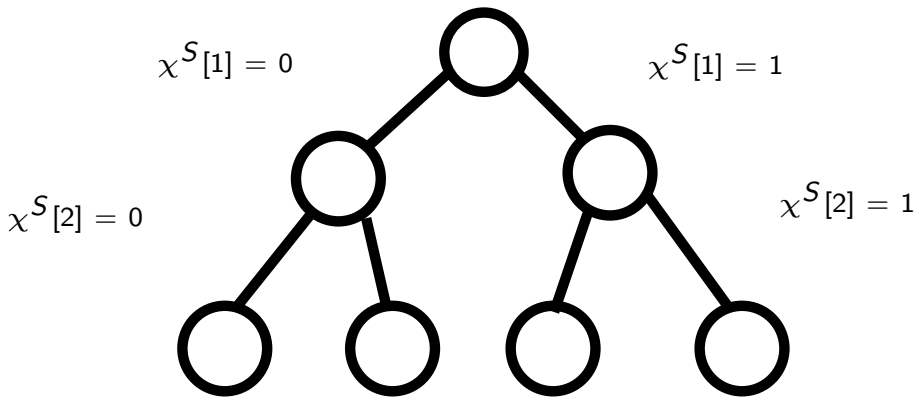
Cet algorithme **glouton** est polynomial (de l'ordre de n^2).

Est-ce que le cas général est lui-aussi polynomial ?

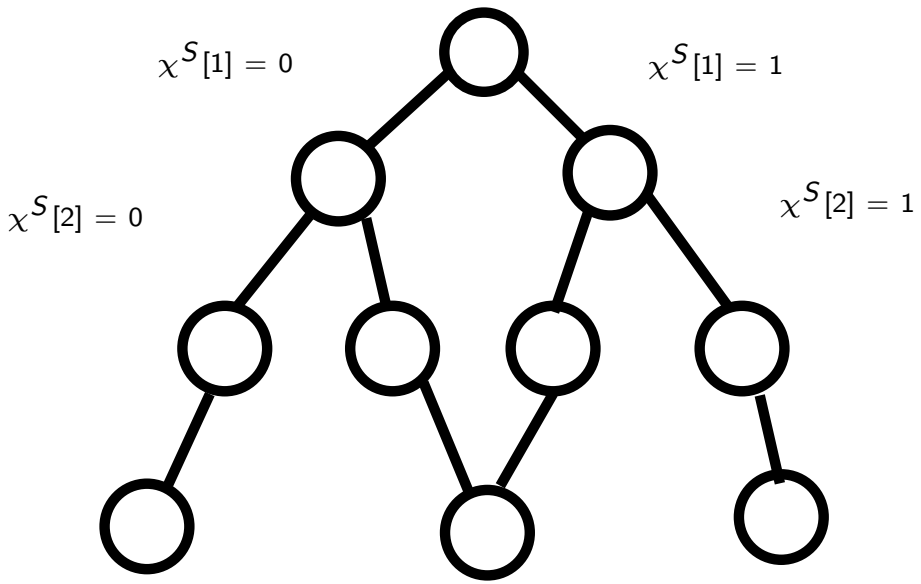
Programmation dynamique



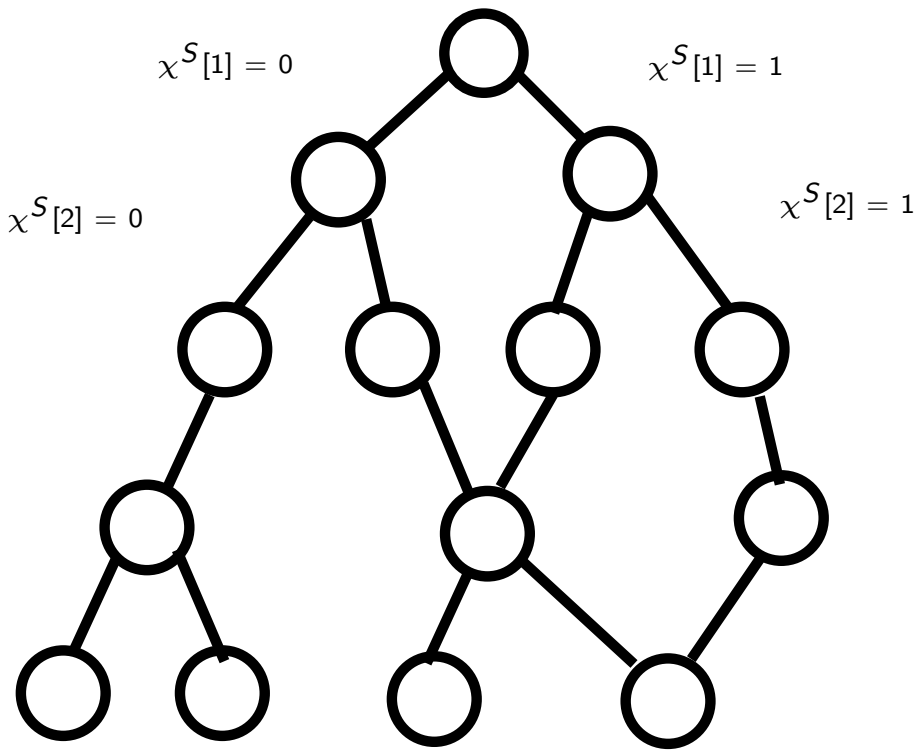
Programmation dynamique



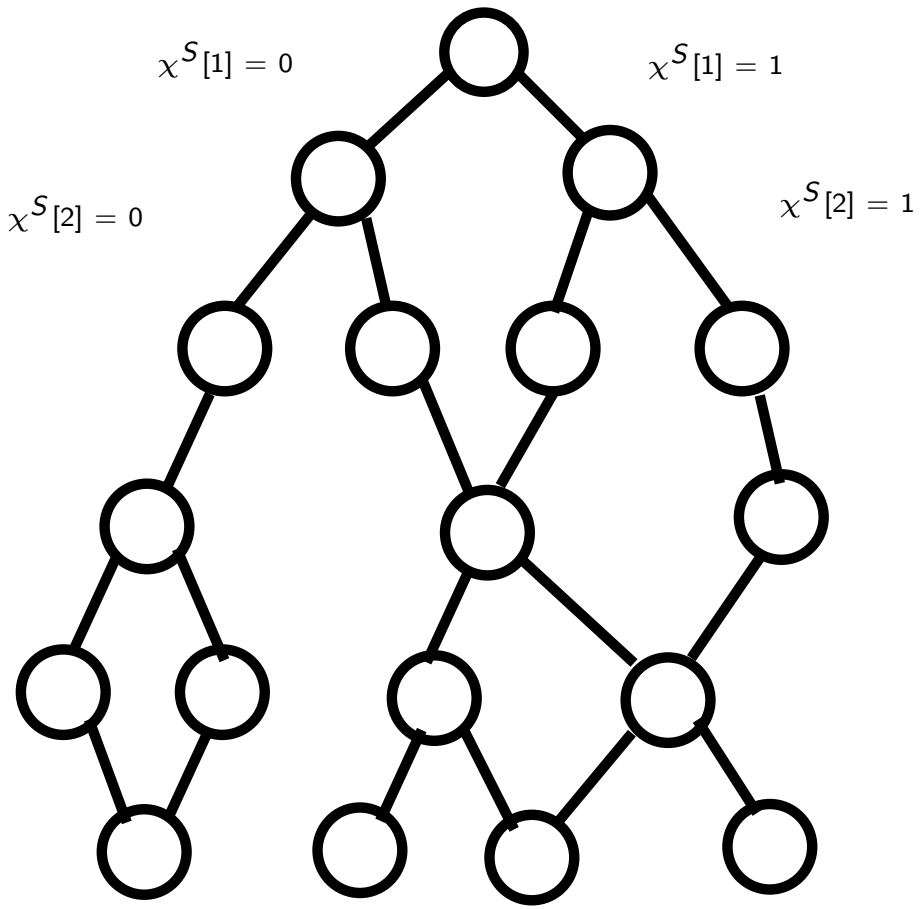
Programmation dynamique



Programmation dynamique



Programmation dynamique



Le "recoupement" freine l'explosion combinatoire

Cas polynomiaux

On connaît beaucoup de cas polynomiaux de problèmes d'optimisation combinatoire :

• Algorithme glouton :
- le problème du sac-à-dos avec tous les objets de même poids.

• Programmation dynamique :
- le problème de sac-à-dos si le sac n'est pas "trop grand"
- le problème du stable dans des graphes simples comme les arbres

• Parcours dans les graphes :
- comment relier tous les points d'un dessin avec une longueur minimale de traits (arbre couvrant minimum)

- ...

$\mathcal{NP}$-difficulté

Dans l'état des connaissances scientifiques, on ne sait pas s'il existe ou non un algorithme polynomial pour résoudre le problème du sac-à-dos en général !

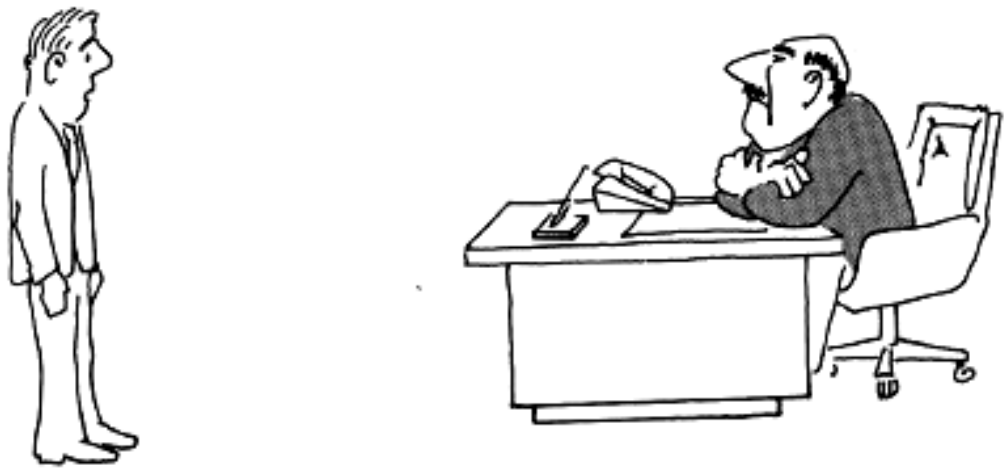
$\mathcal{NP}$-**difficulté**

Dans l'état des connaissances scientifiques, on ne sait pas s'il existe ou non un algorithme polynomial pour résoudre le problème du sac-à-dos en général !

En fait on a pu prouver que ce problème était aussi difficile que tous les problèmes de la classe $\mathcal{NP}$!

On dit qu'un problèmes est $\mathcal{NP}$-**difficile** s'il est aussi difficile que tout problème de la classe $\mathcal{NP}$.

$\mathcal{NP}$-difficulté



Patron, je ne trouve pas d'algorithme polynomial pour le problème du sac-à-dos

Figure prise dans le "Garey et Johnson"

$\mathcal{NP}$-difficulté

Mais si vous pensez que je suis juste pas très doué...

Figure prise dans le "Garey et Johnson"

$\mathcal{NP}$-difficulté

... tous les autres ne le sont pas non plus !

Figure prise dans le "Garey et Johnson"

Problèmes $\mathcal{NP}$-difficiles

On connaît beaucoup de problèmes $\mathcal{NP}$-difficiles :

- le problème du sac-à-dos

- le problème du stable maximum

- les problèmes d'ordonnancement de tâches

- les problèmes de coloration de graphes

- ... et la plupart des problèmes de recherche opérationnelle venant de l'industrie.

Problème du plus court chemin

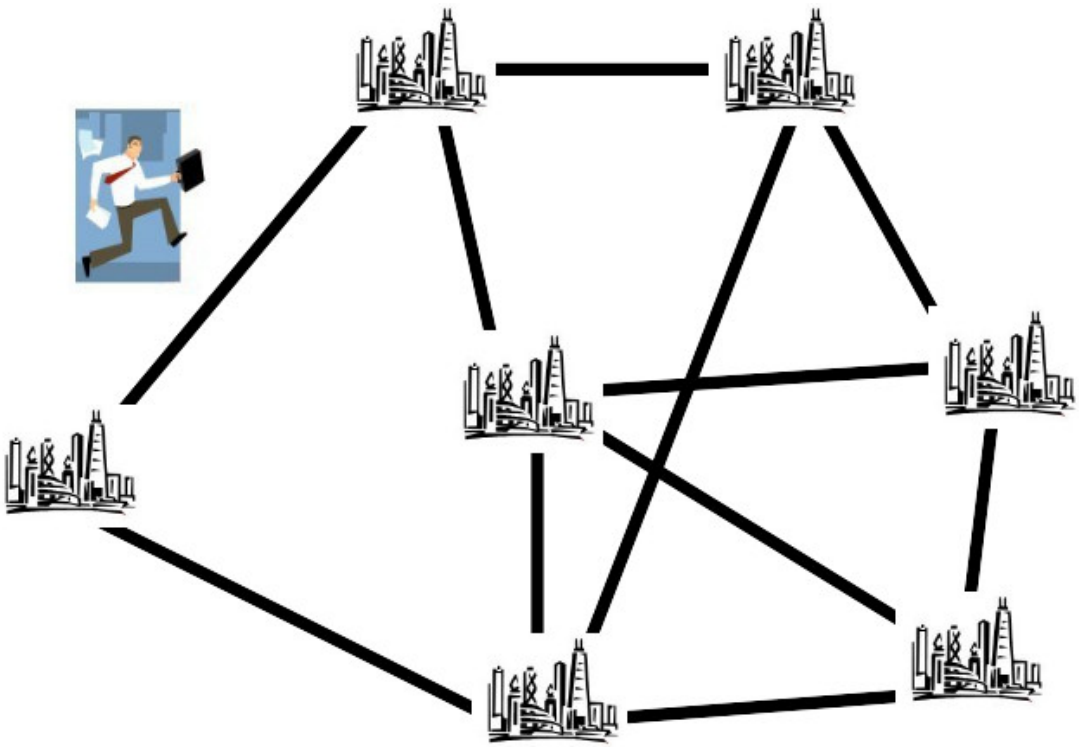
Aller d'un point à un autre au plus court sur une carte.

Problème du plus court chemin

Aller d'un point à un autre au plus court sur une carte.

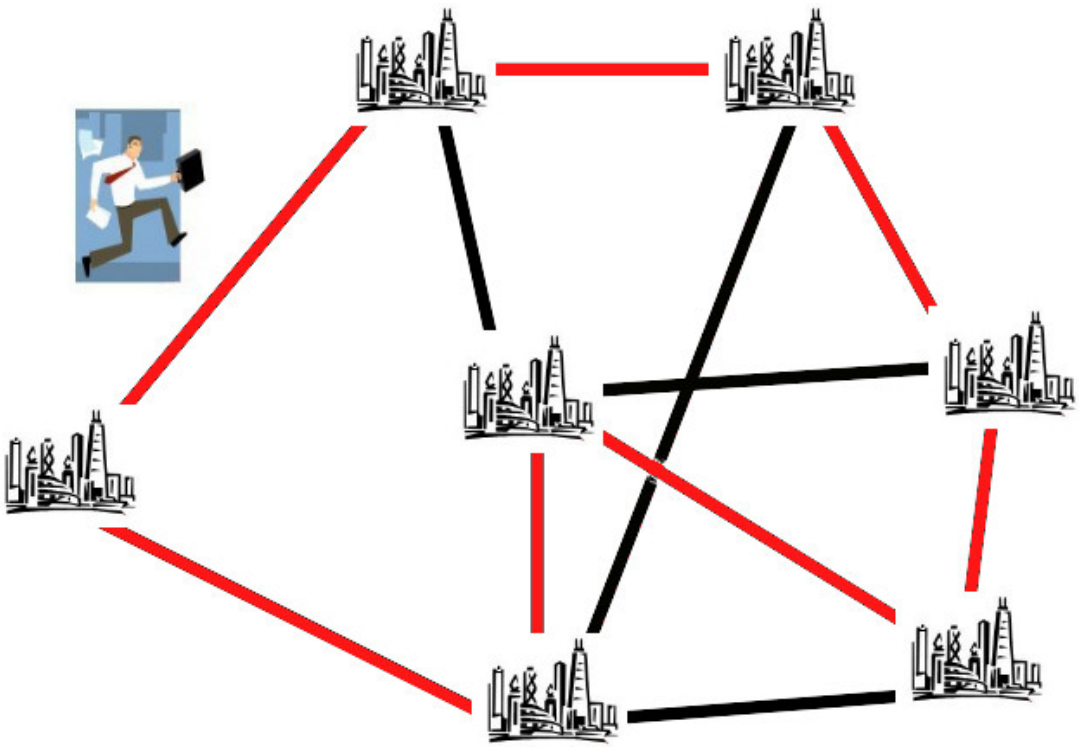
Quelle difficulté ?

Problème du voyageur de commerce



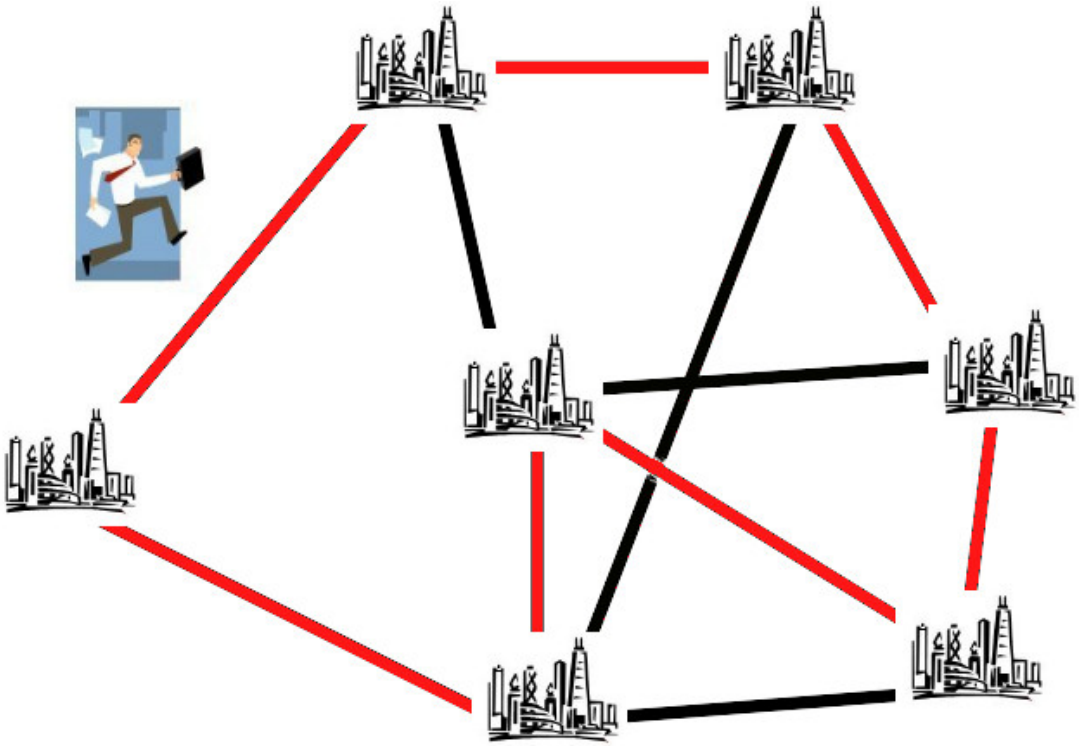
Partir d'un point et y revenir
en passant par tous les points
au plus court sur une carte.

Problème du voyageur de commerce



Partir d'un point et y revenir
en passant par tous les points
au plus court sur une carte.

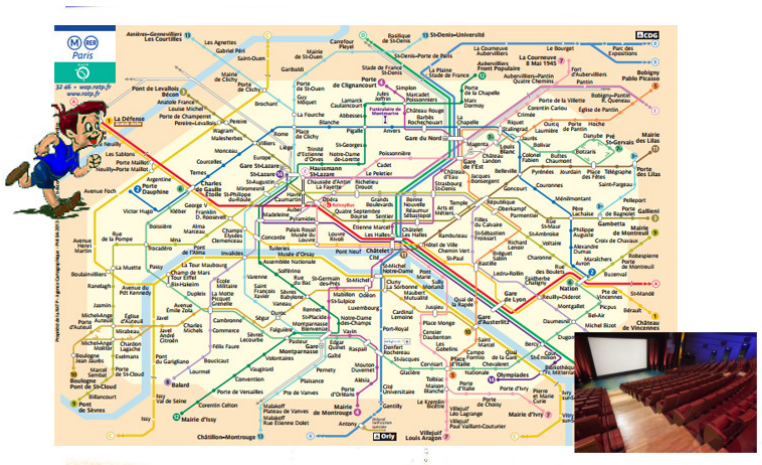
Problème du voyageur de commerce



Partir d'un point et y revenir
en passant par tous les points
au plus court sur une carte.

Quelle difficulté ?

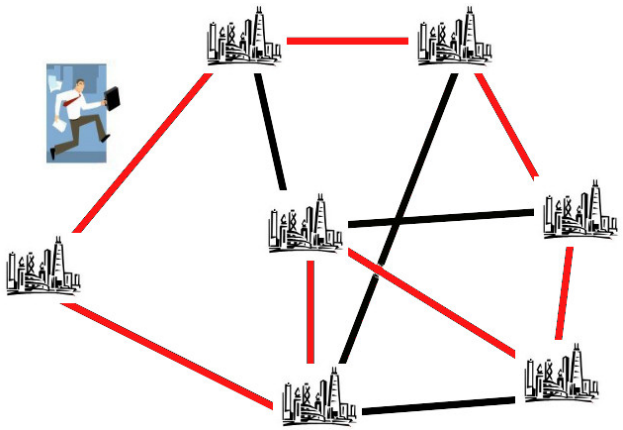
Plus court chemin



Polynomial

On sait le résoudre pour des millions de croisements !

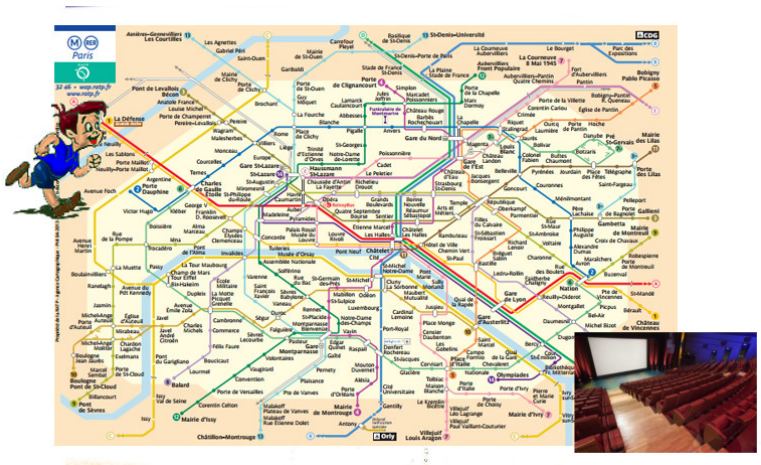
Voyageur de commerce



$\mathcal{NP}$-difficile

Il y a 20 ans, on ne savait pas dépasser quelques centaines de villes !

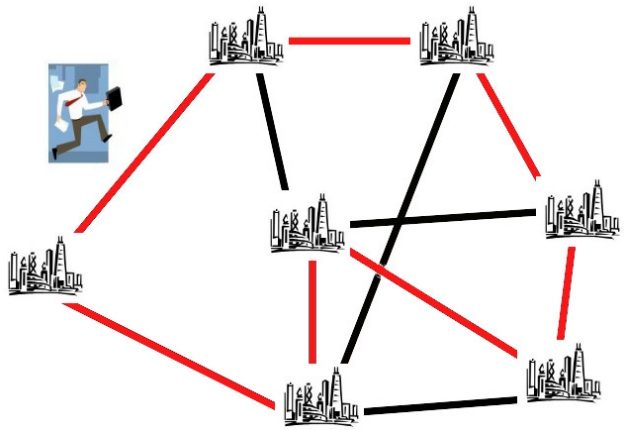
Plus court chemin



Polynomial

On sait le résoudre pour des millions de croisements !

Voyageur de commerce



$\mathcal{NP}$-difficile

Il y a 20 ans, on ne savait pas dépasser quelques centaines de villes !

Et pourtant, en 2003, des instances à 200 000 villes ont été résolues !

Un petit problème tiré d'un livre de lycée

Un fabriquant de yaourt produit 2 types A et B de yaourts à la fraise à partir de fraises, de lait et de sucre.
Chaque yaourt doit respecter les proportions suivantes de matières premières.

	A	B
Fraises	2	1
Lait	1	2
Sucre	0	1

Matières premières en quantité limitée :
- fraises : 800kg, lait : 700kg
- sucre : 300kg.

La vente des yaourts rapportent
- A : 4€ /kg et B : 5€ /kg

Un petit problème tiré d'un livre de lycée

Un fabriquant de yaourt produit 2 types A et B de yaourts à la fraise à partir de fraises, de lait et de sucre.
Chaque yaourt doit respecter les proportions suivantes de matières premières.

	A	B
Fraises	2	1
Lait	1	2
Sucre	0	1

Matières premières en quantité limitée :
fraises : 800kg, lait : 700kg
sucre : 300kg.
La vente des yaourts rapportent
A : 4€ /kg et B : 5€ /kg

Modélisons

x_A quantité en kg de type A à produire
x_B quantité en kg de type B à produire

Un petit problème tiré d'un livre de lycée

Un fabriquant de yaourt produit 2 types A et B de yaourts à la fraise à partir de fraises, de lait et de sucre.
Chaque yaourt doit respecter les proportions suivantes de matières premières.

	A	B
Fraises	2	1
Lait	1	2
Sucre	0	1

Matières premières en quantité limitée :
fraises : 800kg, lait : 700kg
sucre : 300kg.
La vente des yaourts rapportent
A : 4€ /kg et B : 5€ /kg

Modélisons

x_A quantité en kg de type A à produire
x_B quantité en kg de type B à produire

$$\text{Min} \quad 4x_A \quad + \quad 5x_b$$

Un petit problème tiré d'un livre de lycée

Un fabriquant de yaourt produit 2 types A et B de yaourts à la fraise à partir de fraises, de lait et de sucre.
Chaque yaourt doit respecter les proportions suivantes de matières premières.

	A	B
Fraises	2	1
Lait	1	2
Sucre	0	1

Matières premières en quantité limitée :
- fraises : 800kg, lait : 700kg
- sucre : 300kg.

La vente des yaourts rapportent
- A : 4€ /kg et B : 5€ /kg

Modélisons

x_A quantité en kg de type A à produire
x_B quantité en kg de type B à produire

$$
\begin{array}{lrcrcl}
\text{Min} & 4x_A & + & 5x_b & & \\
& \frac{2}{3}x_A & + & \frac{1}{4}x_B & \leq & 800 \\
& \frac{1}{3}x_A & + & \frac{1}{2}x_B & \leq & 700 \\
& & & \frac{1}{4}x_B & \leq & 300 \\
& x_A \geq 0 & & & & \\
& & & x_B \geq 0 & &
\end{array}
$$

Un petit problème tiré d'un livre de lycée

Un fabriquant de yaourt produit 2 types A et B de yaourts à la fraise à partir de fraises, de lait et de sucre.
Chaque yaourt doit respecter les proportions suivantes de matières premières.

	A	B
Fraises	2	1
Lait	1	2
Sucre	0	1

Matières premières en quantité limitée :
- fraises : 800kg, lait : 700kg
- sucre : 300kg.

La vente des yaourts rapportent
- A : 4€ /kg et B : 5€ /kg

Modélisons

x_A quantité en kg de type A à produire
x_B quantité en kg de type B à produire

$$\begin{array}{lllll} \text{Min} & 4x_A & + & 5x_b & \\ & \frac{2}{3}x_A & + & \frac{1}{4}x_B & \leq 800 \\ & \frac{1}{3}x_A & + & \frac{1}{2}x_B & \leq 700 \\ & & & \frac{1}{4}x_B & \leq 300 \\ & x_A \geq 0 & & & \\ & & & x_B \geq 0 & \end{array}$$

C'est un **Programme linéaire**

La programmation linéaire

Programme linéaire :
Optimiser une fonction linéaire
sous la contrainte de respecter
des inégalités linéaires.

$$
\begin{array}{lllll}
\text{Min} & 4x_A & + & 5x_b & \\
 & \frac{2}{3}x_A & + & \frac{1}{4}x_B & \leq 800 \\
 & \frac{1}{3}x_A & + & \frac{1}{2}x_B & \leq 700 \\
 & & & \frac{1}{4}x_B & \leq 300 \\
 & x_A \geq 0 & & & \\
 & & & x_B \geq 0 &
\end{array}
$$

La programmation linéaire

Programme linéaire :
Optimiser une fonction linéaire sous la contrainte de respecter des inégalités linéaires.

$$
\begin{array}{lrcrcl}
\text{Min} & 4x_A & + & 5x_b & & \\
 & \frac{2}{3}x_A & + & \frac{1}{4}x_B & \leq & 800 \\
 & \frac{1}{3}x_A & + & \frac{1}{2}x_B & \leq & 700 \\
 & & & \frac{1}{4}x_B & \leq & 300 \\
 & x_A \geq 0 & & & & \\
 & & & x_B \geq 0 & &
\end{array}
$$

On peut résoudre tout programme linéaire **en temps polynomial !**.
Il existe des algorithmes efficaces (simplexe, points intérieurs...).

La programmation linéaire

Programme linéaire :
Optimiser une fonction linéaire sous la contrainte de respecter des inégalités linéaires.

$$
\begin{array}{lrcrcl}
\text{Min} & 4x_A & + & 5x_b & & \\
 & \frac{2}{3}x_A & + & \frac{1}{4}x_B & \leq & 800 \\
 & \frac{1}{3}x_A & + & \frac{1}{2}x_B & \leq & 700 \\
 & & & \frac{1}{4}x_B & \leq & 300 \\
 & x_A \geq 0 & & & & \\
 & & & x_B \geq 0 & &
\end{array}
$$

On peut résoudre tout programme linéaire **en temps polynomial !**.
Il existe des algorithmes efficaces (simplexe, points intérieurs...).

Mais

La programmation linéaire

Programme linéaire :
Optimiser une fonction linéaire sous la contrainte de respecter des inégalités linéaires.

$$
\begin{array}{lllll}
\text{Min} & 4x_A & + & 5x_b & \\
 & \frac{2}{3}x_A & + & \frac{1}{4}x_B & \leq 800 \\
 & \frac{1}{3}x_A & + & \frac{1}{2}x_B & \leq 700 \\
 & & & \frac{1}{4}x_B & \leq 300 \\
 & x_A \geq 0 & & & \\
 & & & x_B \geq 0 &
\end{array}
$$

On peut résoudre tout programme linéaire **en temps polynomial !**.
Il existe des algorithmes efficaces (simplexe, points intérieurs...).

Mais

La programmation linéaire est-elle un problème d'optimisation combinatoire ?

Réprésentation graphique

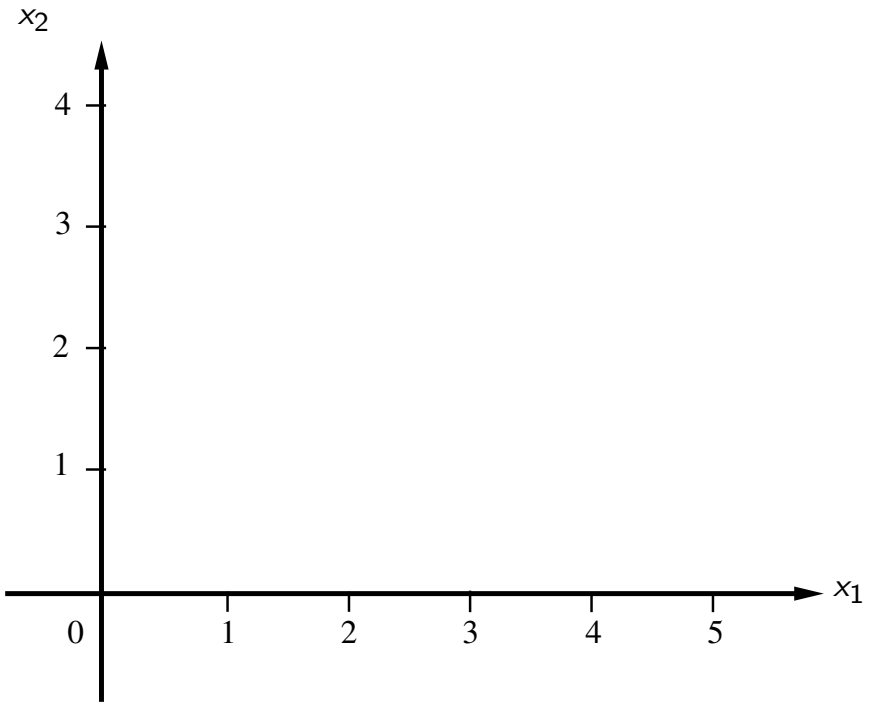
Un programme linéaire à 2 variables
se représente dans un repère à 2 dimensions.

$$
\begin{aligned}
\text{Max } z = \quad & 2x_1 + x_2 \\
& x_1 - 4x_2 \leq 0 \\
& 3x_1 + 4x_2 \leq 15 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Réprésentation graphique

Un programme linéaire à 2 variables
se représente dans un repère à 2 dimensions.

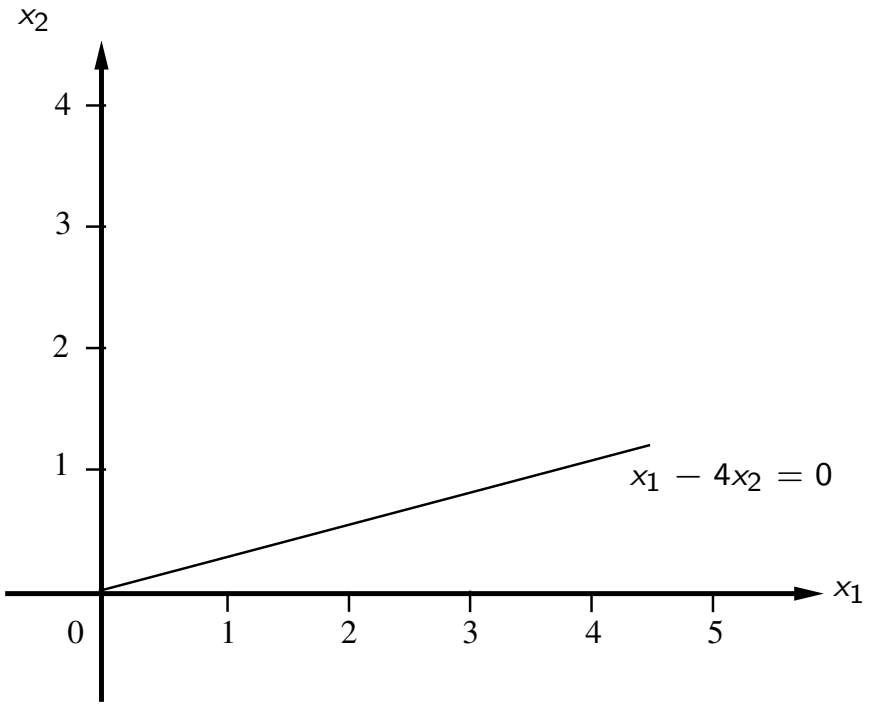
$$\begin{aligned} \text{Max } z = \quad & 2x_1 + x_2 \\ & x_1 - 4x_2 \leq 0 \\ & 3x_1 + 4x_2 \leq 15 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$



Réprésentation graphique

Un programme linéaire à 2 variables
se représente dans un repère à 2 dimensions.

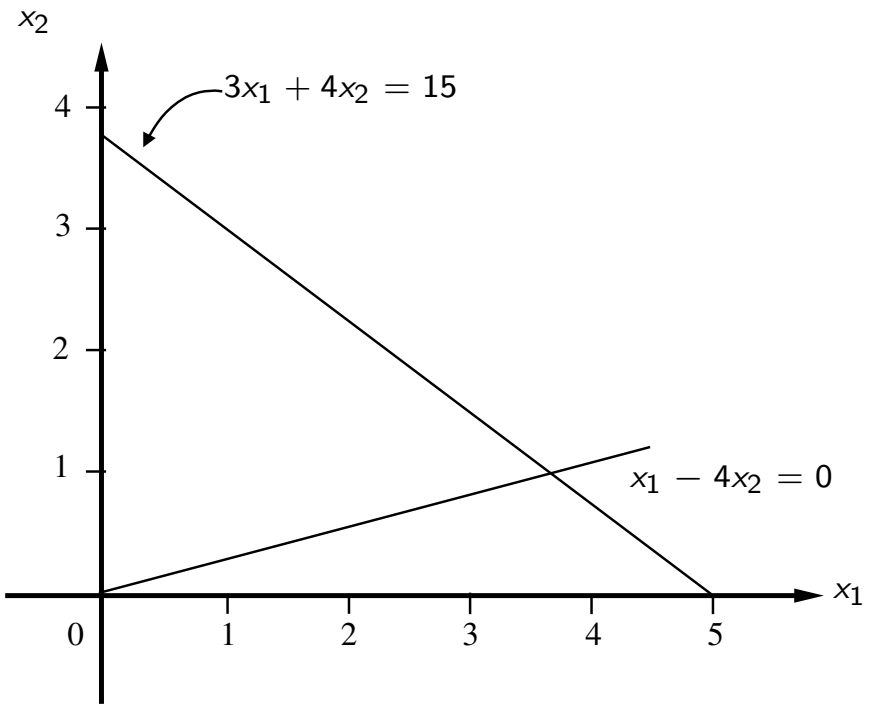
$$
\begin{aligned}
\text{Max } z = \quad & 2x_1 + x_2 \\
& x_1 - 4x_2 \leq 0 \\
& 3x_1 + 4x_2 \leq 15 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$



Réprésentation graphique

Un programme linéaire à 2 variables
se représente dans un repère à 2 dimensions.

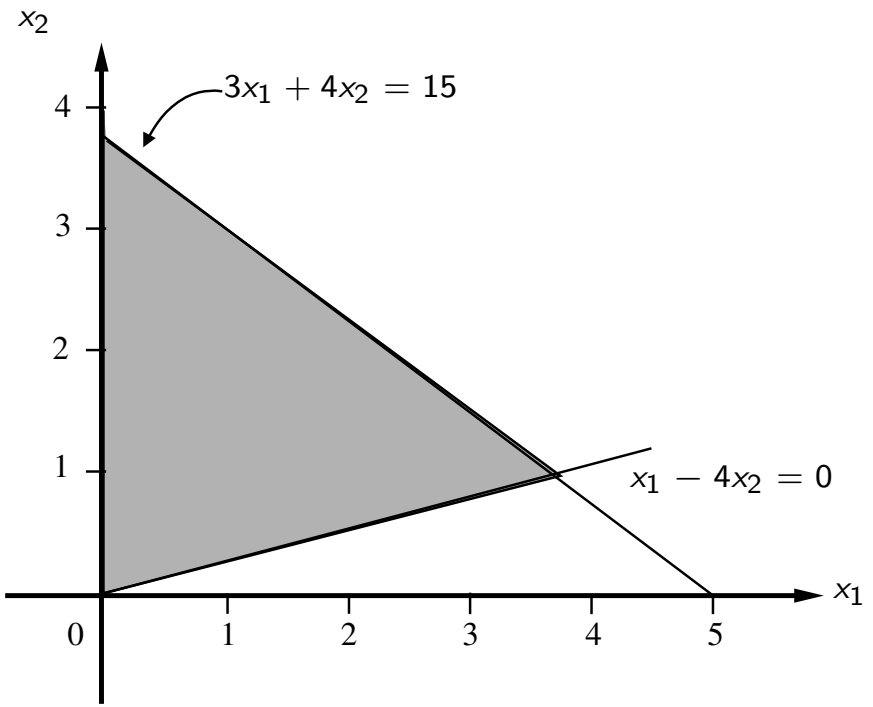
$$\begin{aligned} \text{Max } z = \quad & 2x_1 + x_2 \\ & x_1 - 4x_2 \leq 0 \\ & 3x_1 + 4x_2 \leq 15 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$



Réprésentation graphique

Un programme linéaire à 2 variables
se représente dans un repère à 2 dimensions.

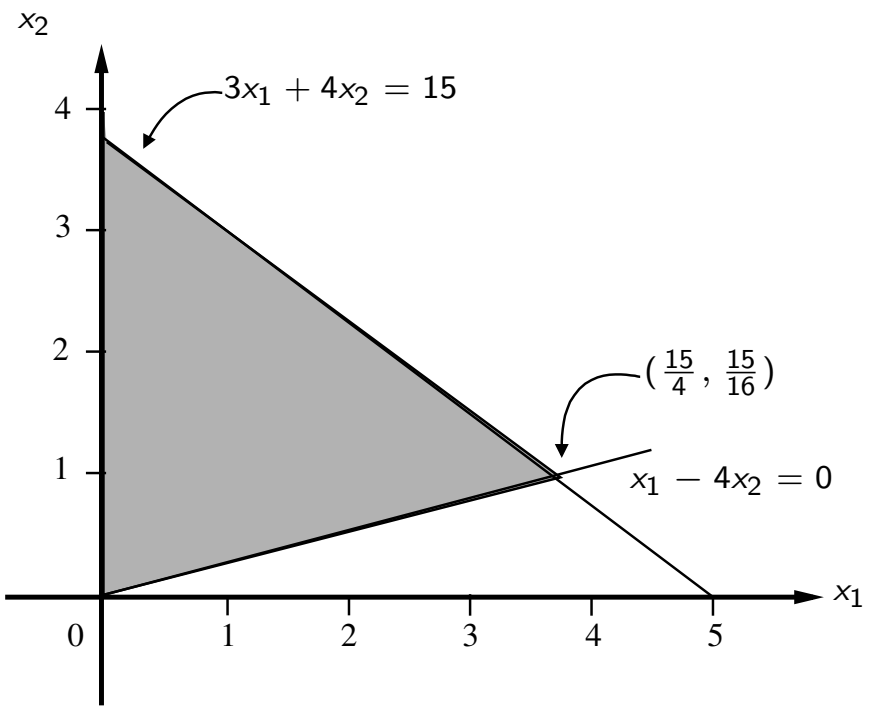
$$\begin{aligned} \text{Max } z = \quad & 2x_1 + x_2 \\ & x_1 - 4x_2 \leq 0 \\ & 3x_1 + 4x_2 \leq 15 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$



Réprésentation graphique

Un programme linéaire à 2 variables
se représente dans un repère à 2 dimensions.

$$\begin{aligned} \text{Max } z = \quad & 2x_1 + x_2 \\ & x_1 - 4x_2 \leq 0 \\ & 3x_1 + 4x_2 \leq 15 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$



Solution optimale
$x_1 = \frac{15}{4}$ $x_2 = \frac{15}{16}$

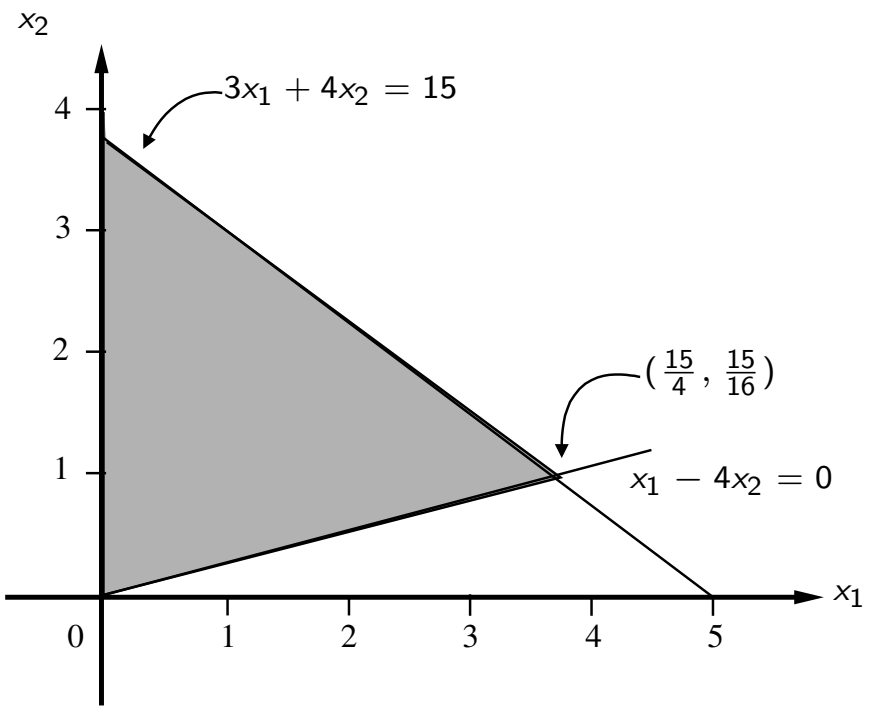
Optimisation Combinatoire ?

Un programme linéaire décrit un ensemble de solutions qui est un **polyèdre**.

Optimisation Combinatoire ?

Un programme linéaire décrit un ensemble de solutions qui est un **polyèdre**.

Pour 2 variables,
un polygone.

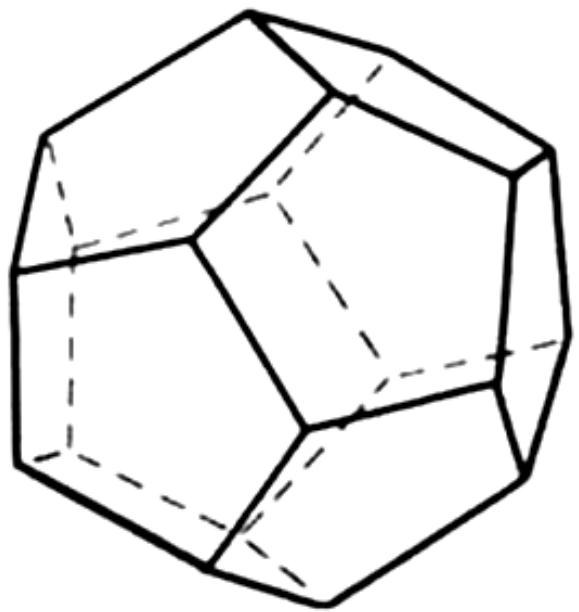


Optimisation Combinatoire ?

Un programme linéaire décrit un ensemble de solutions qui est un **polyèdre**.

Pour 2 variables,
un polygone.

Pour 3 variables,
une figure "3D".



Optimisation Combinatoire ?

Un programme linéaire décrit un ensemble de solutions qui est un **polyèdre**.

Pour 2 variables,
un polygone.

Pour 3 variables,
une figure "3D".

Pour n variables, ...

...

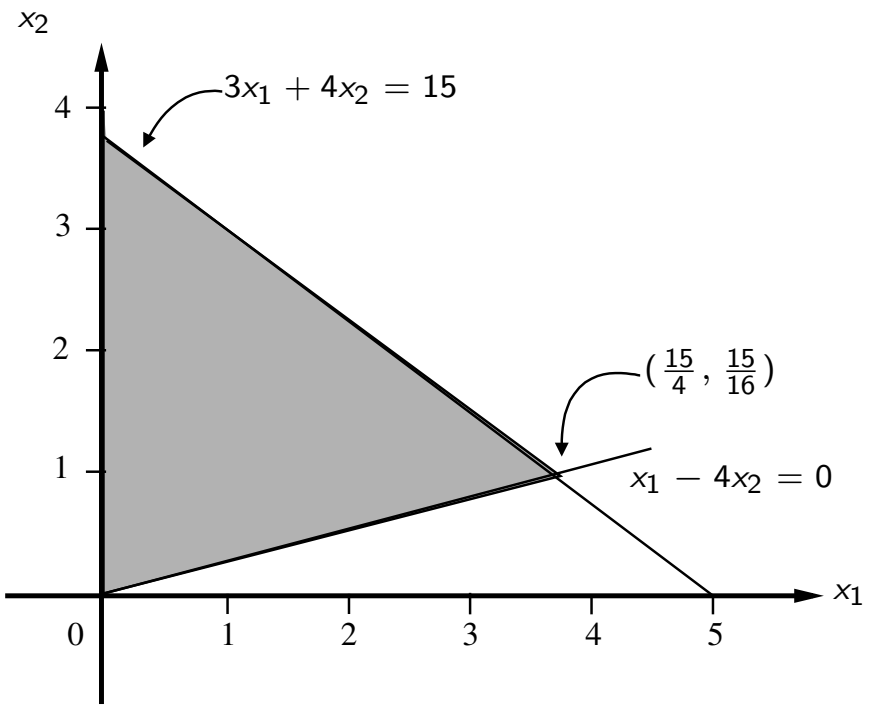
Optimisation Combinatoire ?

Les solutions **optimales**
d'un pogramme linéaire
peuvent être choisies
parmi les **points extrêmes**
qui sont à l'intersection
de n facettes du polyèdre.

Optimisation Combinatoire ?

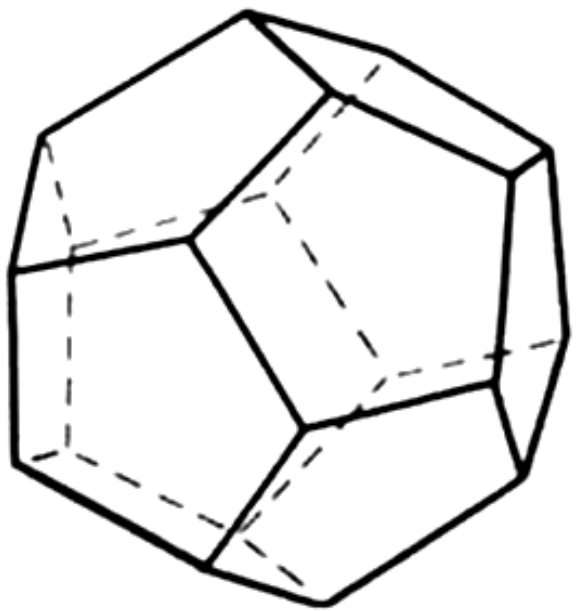
Les solutions **optimales** d'un pogramme linéaire peuvent être choisies parmi les **points extrêmes** qui sont à l'intersection de n facettes du polyèdre.

Pour 2 variables, l'intersection de 2 droites



Optimisation Combinatoire ?

Les solutions **optimales**
d'un pogramme linéaire
peuvent être choisies
parmi les **points extrêmes**
qui sont à l'intersection
de n facettes du polyèdre.



Pour 2 variables,
l'intersection de 2 droites

Pour 3 variables,
l'intersection de 3 facettes

Optimisation Combinatoire ?

Les solutions **optimales**
d'un pogramme linéaire
peuvent être choisies
parmi les **points extrêmes**
qui sont à l'intersection
de n facettes du polyèdre.

Pour un programme linéaire
à n variables
et m inégalités linéaires
et n inégalités $x_i \geq 0$

Optimisation Combinatoire ?

Les solutions **optimales**
d'un pogramme linéaire
peuvent être choisies
parmi les **points extrêmes**
qui sont à l'intersection
de n facettes du polyèdre.

Pour un programme linéaire
à n variables
et m inégalités linéaires
et n inégalités $x_i \geq 0$

Combien y-a-t-il de solutions optimales possibles ?

Optimisation Combinatoire ?

Les solutions **optimales**
d'un pogramme linéaire
peuvent être choisies
parmi les **points extrêmes**
qui sont à l'intersection
de n facettes du polyèdre.

Pour un programme linéaire
à n variables
et m inégalités linéaires
et n inégalités $x_i \geq 0$

Combien y-a-t-il de solutions optimales possibles ?

Au pire autant que de façon de prendre
n inégalités parmi $n + m$:

$$C_n^{n+m} = \frac{(n+m)!}{n!m!}$$

Nombre exponentiel de solutions dont certaines sont optimales.

Optimisation Combinatoire ?

Les solutions **optimales**
d'un pogramme linéaire
peuvent être choisies
parmi les **points extrêmes**
qui sont à l'intersection
de n facettes du polyèdre.

Pour un programme linéaire
à n variables
et m inégalités linéaires
et n inégalités $x_i \geq 0$

Combien y-a-t-il de solutions optimales possibles ?

Au pire autant que de façon de prendre
n inégalités parmi $n + m$:

$$C_n^{n+m} = \frac{(n+m)!}{n!m!}$$

Nombre exponentiel de solutions dont certaines sont optimales.

C'est bien un problème d'optimisation combinatoire !

Programmation linéaire en nombres entiers

En obligeant les variables à être entières ou en 0/1, on définit un **Programme Linéaire en Nombres Entiers (PLNE)** qui permet de formuler les problèmes d'Optimisation Combinatoire.

Problème du sac-à-dos

$$\text{Max} \sum_{i=1}^{n} c_i x_i$$

$$\sum_{i=1}^{n} a_i x_i \leq b,$$

$$0 \leq x_i \leq 1, \text{ pour tout objet } i = 1, ..., n,$$

$$x_i \in \mathbb{N}, \text{ pour tout objet } i = 1, ..., n.$$

Programmation linéaire en nombres entiers

En obligeant les variables à être entières ou en 0/1, on définit un **Programme Linéaire en Nombres Entiers (PLNE)** qui permet de formuler les problèmes d'Optimisation Combinatoire.

Problème du stable maximum

$$\begin{aligned} \text{Max} \sum_{u \in V} c(u)x(u) & \\ x(u) + x(v) \leq 1, & \quad \text{pour toute arête } uv, \\ 0 \leq x(u) \leq 1 & \quad \text{pour tout sommet } u, \\ x(u) \in \{0,1\}, & \quad \text{pour tout sommet } u. \end{aligned}$$

Programmation linéaire en nombres entiers

En obligeant les variables à être entières ou en 0/1, on définit un **Programme Linéaire en Nombres Entiers (PLNE)** qui permet de formuler les problèmes d'Optimisation Combinatoire.

Problème du stable maximum

$$\begin{aligned} \text{Max} \sum_{u \in V} c(u)x(u) & \\ x(u) + x(v) \leq 1, & \quad \text{pour toute arête } uv, \\ 0 \leq x(u) \leq 1 & \quad \text{pour tout sommet } u, \\ x(u) \in \{0,1\}, & \quad \text{pour tout sommet } u. \end{aligned}$$

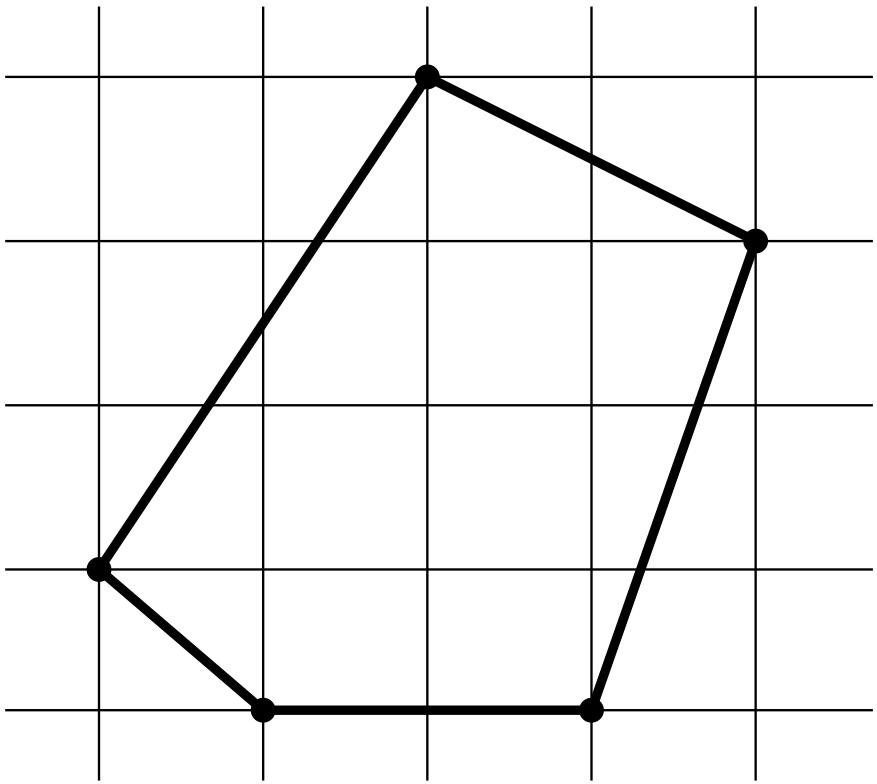
Donc la PLNE est $\mathcal{NP}$-difficile

Se ramener à la Programmation linéaire ?

Une idée qui peut paraître naturelle est de ramener un problème combinatoire à un programme linéaire...

Se ramener à la Programmation linéaire ?

Une idée qui peut paraître naturelle est de ramener un problème combinatoire à un programme linéaire...



à un programme linéaire correspondant à un **polyèdre entier** dont tous les points extrêmes sont entiers

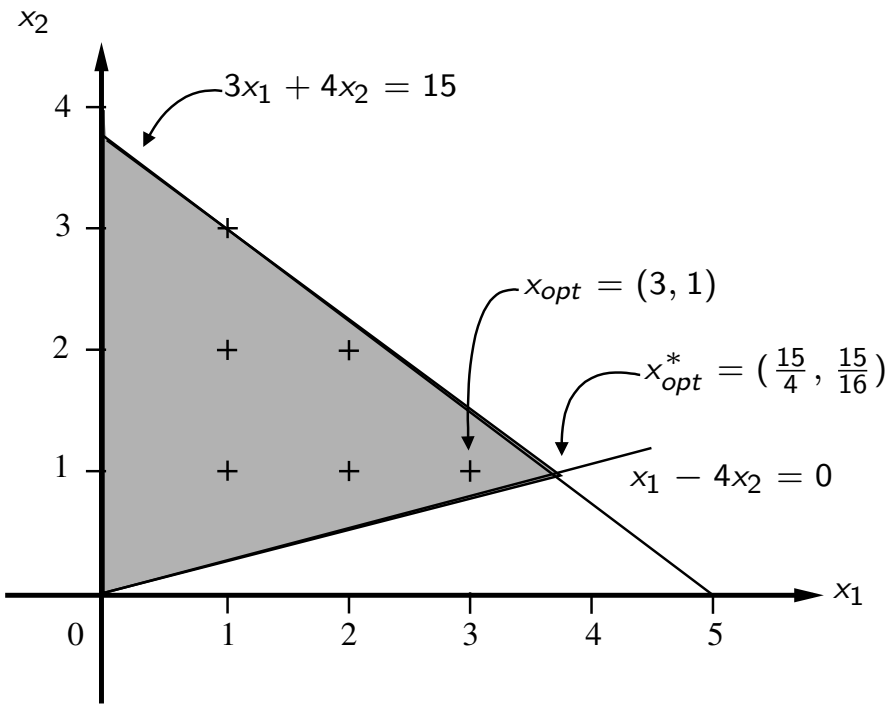
et dont les solutions sont entières !

Soit le PLNE (P)

$$\begin{aligned}
&\text{Max } z = 2x_1 + x_2 \\
&x_1 - 4x_2 \leq 0, \\
&3x_1 + 4x_2 \leq 15, \\
&x_1 \geq 0, \\
&x_2 \geq 0, \\
&x_1, x_2 \in \boldsymbol{N}.
\end{aligned}$$

Soit le PLNE (P)

$$\text{Max } z = 2x_1 + x_2$$
$$x_1 - 4x_2 \leq 0,$$
$$3x_1 + 4x_2 \leq 15,$$
$$x_1 \geq 0,$$
$$x_2 \geq 0,$$
$$x_1, x_2 \in \boldsymbol{N}.$$



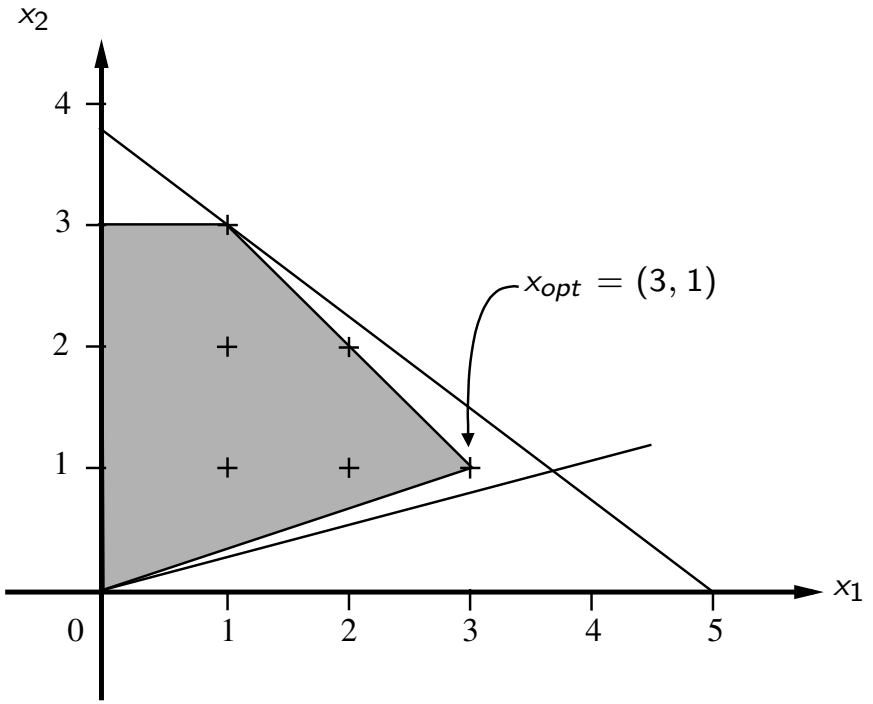
x_{opt} solution entière optimale de (P).
x^*_{opt} solution "fractionnaire" optimale de (P^*).

Regardons les points extrêmes qui sont entiers par construction).

$$\begin{aligned}
&\text{Max } z = 2x_1 + x_2 \\
&x_1 - 4x_2 \leq 0, \\
&3x_1 + 4x_2 \leq 15, \\
&x_1 \geq 0, \\
&x_2 \geq 0, \\
&x_1, x_2 \in \mathbf{N}.
\end{aligned}$$

Regardons les points extrêmes qui sont entiers par construction).

$$\text{Max } z = 2x_1 + x_2$$
$$x_1 - 4x_2 \leq 0,$$
$$3x_1 + 4x_2 \leq 15,$$
$$x_1 \geq 0,$$
$$x_2 \geq 0,$$
$$x_1, x_2 \in \boldsymbol{N}.$$



On prend le **plus petit polyèdre** contenant tous les points entiers du programme linéaire : il s'appelle enveloppe convexe des solutions.

Enveloppe convexe :

Notons $\mathcal{S}$ l'ensemble des solutions du problème d'optimisation combinatoire $\mathcal{P}$.
Donc

$$\mathcal{P} \quad \text{Max } \{cx \mid x \in \mathcal{S}\}$$

Enveloppe convexe :

Notons $\mathcal{S}$ l'ensemble des solutions du problème d'optimisation combinatoire $\mathcal{P}$.
Donc

$$\mathcal{P} \quad \text{Max } \{cx \mid x \in \mathcal{S}\}$$

Notons l'enveloppe convexe $conv(\mathcal{S})$ des solutions de $\mathcal{P}$.

$$\mathcal{P} \quad \text{Max } \{cx \mid x \in conv(\mathcal{S})\}.$$

Enveloppe convexe :

Notons $\mathcal{S}$ l'ensemble des solutions du problème d'optimisation combinatoire $\mathcal{P}$.
Donc

$$\mathcal{P} \quad \text{Max } \{cx \mid x \in \mathcal{S}\}$$

Notons l'enveloppe convexe $conv(\mathcal{S})$ des solutions de $\mathcal{P}$.

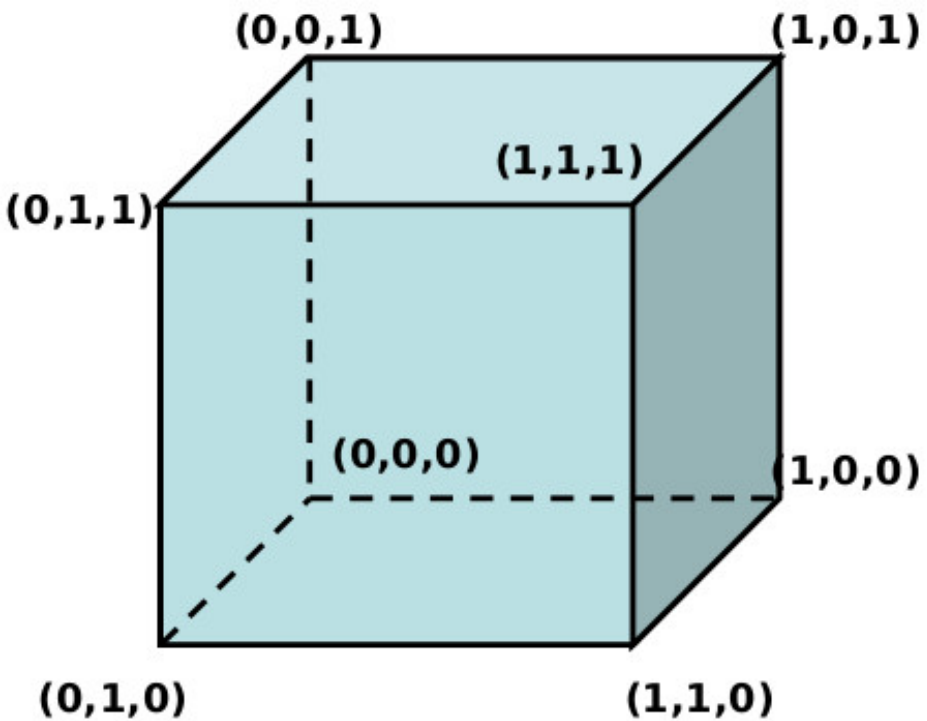
$$\mathcal{P} \quad \text{Max } \{cx \mid x \in conv(\mathcal{S})\}.$$

Or une enveloppe convexe est un programme linéaire !

Si l'on connaissait les inégalités définissant $conv(\mathcal{S})$, le problème serait résolue par l'algo du simplexe ou un algo de coupes.

Un exemple en dimension 3 :

Prenons l'ensemble des vecteurs d'incidence des ensembles de 3 objets du sac-à-dos. Quelle est leur enveloppe convexe ?

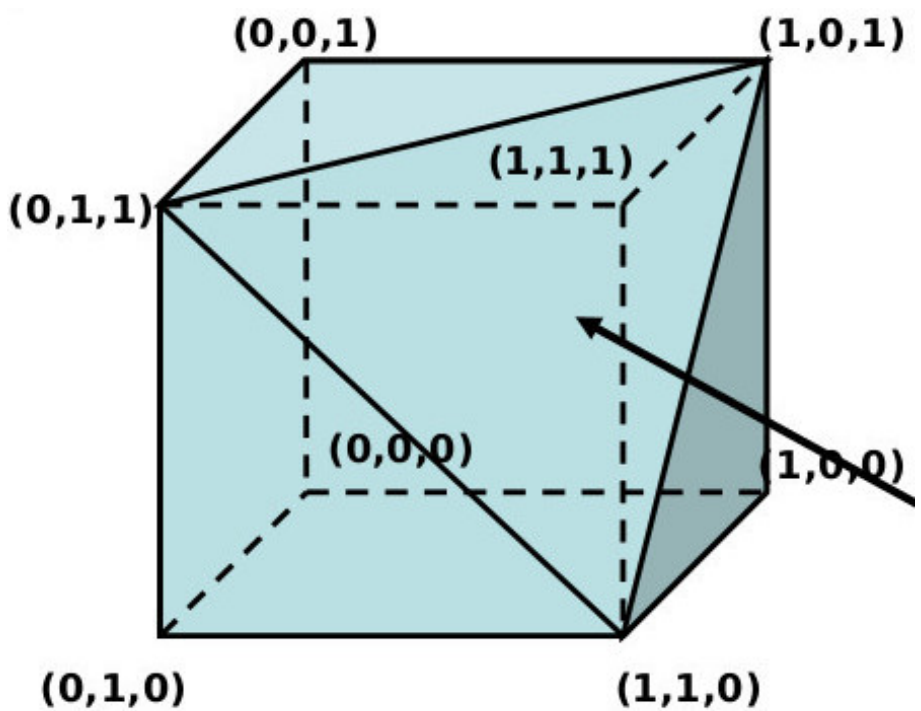


Elle est incluse dans l'hypercube de dimension 3 :

$$
\begin{aligned}
x_1 &\leq 1 \\
x_2 &\leq 1 \\
x_3 &\leq 1 \\
x_1 &\geq 0 \\
x_2 &\geq 0 \\
x_3 &\geq 0
\end{aligned}
$$

Un exemple en dimension 3 :

Prenons l'ensemble des vecteurs d'incidence des sous-ensembles de sommets induisant un sous-graphe sans cycle dans un cycle de 3 sommets. Quelle est leur enveloppe convexe ?

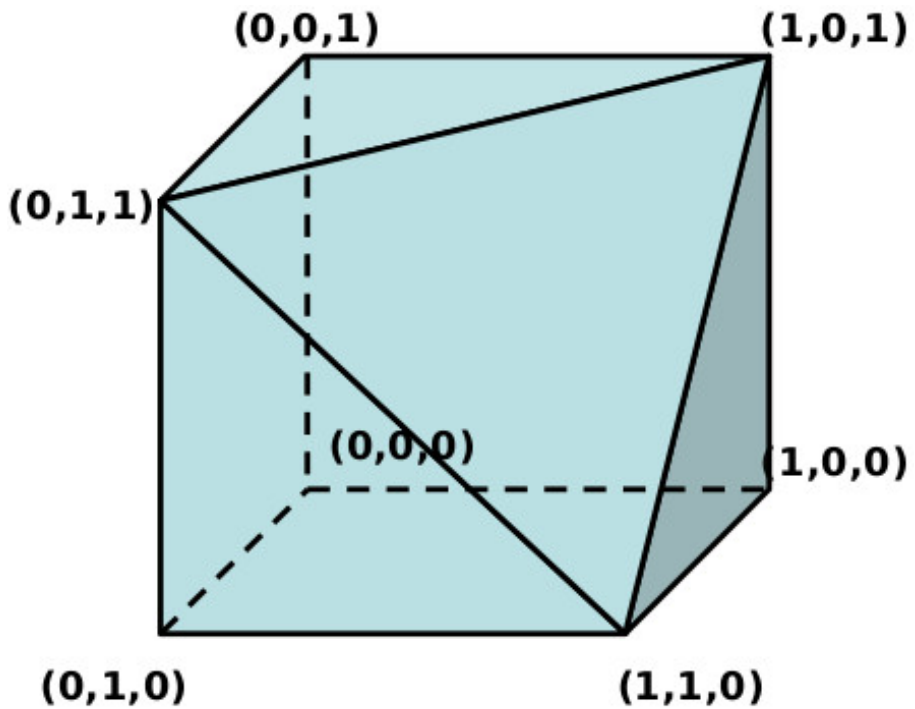


Dans ce sac-à-dos, on ne peut pas prendre 3 objets en même temps

$$x_1 + x_2 + x_3 \leq 2$$

Un exemple en dimension 3 :

Prenons l'ensemble des vecteurs d'incidence des sous-ensembles de sommets induisant un sous-graphe sans cycle dans un cycle de 3 sommets. Quelle est leur enveloppe convexe ?



L'enveloppe est exactement donnée :

$$
\begin{aligned}
x_1 + x_2 + x_3 &\leq 2 \\
x_1 &\leq 1 \\
x_2 &\leq 1 \\
x_3 &\leq 1 \\
x_1 &\geq 0 \\
x_2 &\geq 0 \\
x_3 &\geq 0
\end{aligned}
$$

Mais...

A moins que P=NP, on ne peut pas connaître toutes les inégalités définissant l'enveloppe convexe d'un problème difficile !

Donc on ne peut ramener un problème d'Optimisation Combinatoire à un simple programme linéaire...

Et pourtant !

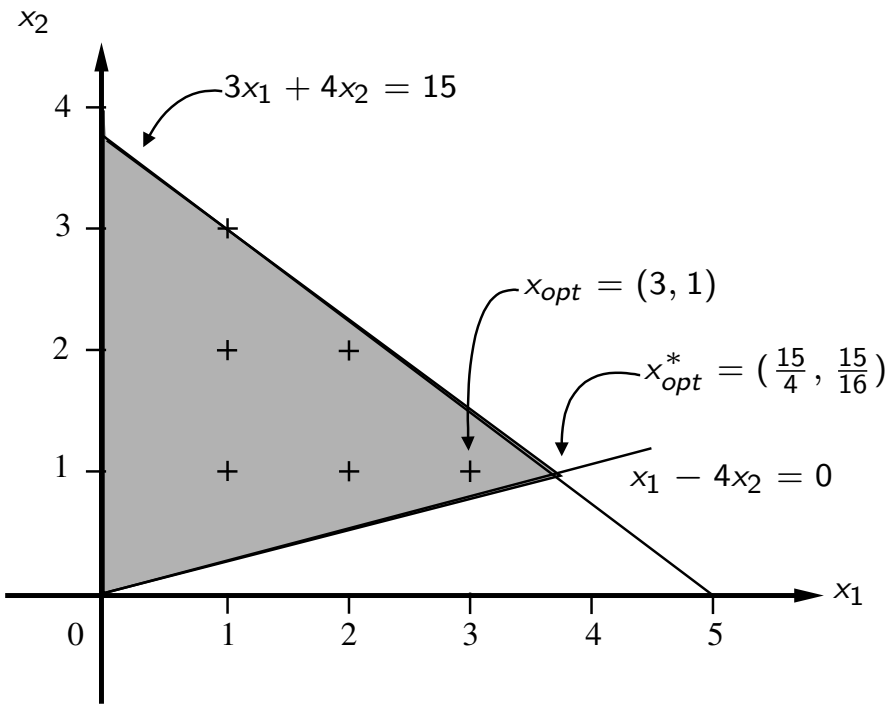
Prenons la **relaxation linéaire d'un programme linéaire** où on ôte l'obligation d'être entier.

$$\begin{aligned}
&\text{Max } z = 2x_1 + x_2 \\
&x_1 - 4x_2 \leq 0, \\
&3x_1 + 4x_2 \leq 15, \\
&x_1 \geq 0, \\
&x_2 \geq 0, \\
&x_1, x_2 \in \mathbb{N}.
\end{aligned}$$

Et pourtant !

Prenons la **relaxation linéaire d'un programme linéaire** où on ôte l'obligation d'être entier.

$$\begin{aligned}
&\text{Max } z = 2x_1 + x_2 \\
&x_1 - 4x_2 \leq 0, \\
&3x_1 + 4x_2 \leq 15, \\
&x_1 \geq 0, \\
&x_2 \geq 0, \\
&x_1, x_2 \in \mathbb{N}.
\end{aligned}$$



La solution x^*_{opt} solution "fractionnaire" est une borne supérieure.

Elagage

Lors de l'énumération, on peut utiliser cette borne supérieure :

Elagage : Ne pas explorer un nœud si l'évaluation de ce sous-problème est inférieure à la meilleure solution connue !

Plus la borne est précise, plus l'explosion combinatoire de l'éunumération est freinée.

Renforcer la formulation

Une étude algébrique et algorithmique des inégalités linéaires de l'enveloppe convexe des solutions permet une **connaissance partielle** du polyèdre du problème.
$\rightarrow$ En ajoutant ces inégalités, on renforce la borne et on permet d'améliorer l'élagage pendant l'énumération.

Renforcer la formulation

Une étude algébrique et algorithmique des inégalités linéaires de l'enveloppe convexe des solutions permet une **connaissance partielle** du polyèdre du problème.
$\rightarrow$ En ajoutant ces inégalités, on renforce la borne et on permet d'améliorer l'élagage pendant l'énumération.

De plus ces inégalités permettent d'arêter l'énumération en trouvant souvent des cas **entiers** où l'on connaît alors une solution de bonne valeur en cours d'énumération.
$\rightarrow$ Si la meilleure borne supérieure est la valeur de cette solution, on a alors trouvé la solution optimale : on stoppe alors l'énumération.

Le problème du voyageur de commerce

Après une 40 aine d'années de recherche sur l'enveloppe convexe des solutions du problème de voyageur de commerce, on a appris beaucoup sur les inégalités de polyèdre.

En plongeant ces résultats provenant de chercheurs travaillant un peu partout sur la planète, une équipe de chercheurs a réussi à implémenter un algorithme de branchement enrichi par toutes ces inégalités.

Des instances de plusieurs centaines de milliers de villes ont été résolues (même une d'un million !).

Les solveurs PLNE

Avec des idées similaires, les solveurs PLNE commerciaux (Cplex, Gurobi) ou universitaires (SCIP) ont connu des améliorations gigantesques !

Ils s'améliorent peu à peu en s'appuyant sur les résultats des chercheurs repoussant sans cesse la limite des problèmes de $\mathcal{NP}$ résolus pour des tailles de plus en plus grandes.

Les solveurs PLNE

Avec des idées similaires, les solveurs PLNE commerciaux (Cplex, Gurobi) ou universitaires (SCIP) ont connu des améliorations gigantesques !

Ils s'améliorent peu à peu en s'appuyant sur les résultats des chercheurs repoussant sans cesse la limite des problèmes de $\mathcal{NP}$ résolus pour des tailles de plus en plus grandes.

Mais on ne sait toujours pas résoudre le problème du stable pour un millier de sommets... et bien d'autres problèmes...