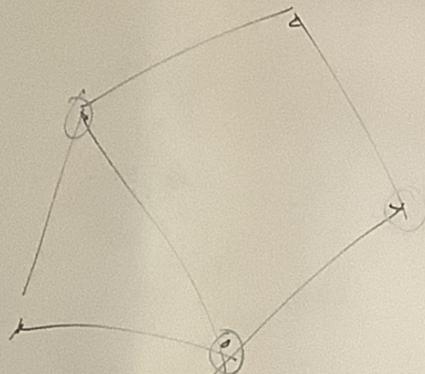




Séance 9

Exercice 4.5.



Soit X un recouvrement
et U un ensemble d'arêtes disjoints

On sait que pour toute arête $\{x, y\}$ de G
 x ou y sont dans X

\Rightarrow chaque arête de U a au
moins une extrémité dans X

Soit pour une arête $a \in U$
 $x(a)$ une extrémité de a dans X

1
Donne les arêtes de U sont disjointes

si $a \in U$ 2 arête $\cdot a, a'$

$$x(a) \neq x(a')$$

\Rightarrow les $x(a)$ sont tous distincts par
 $a \in U$

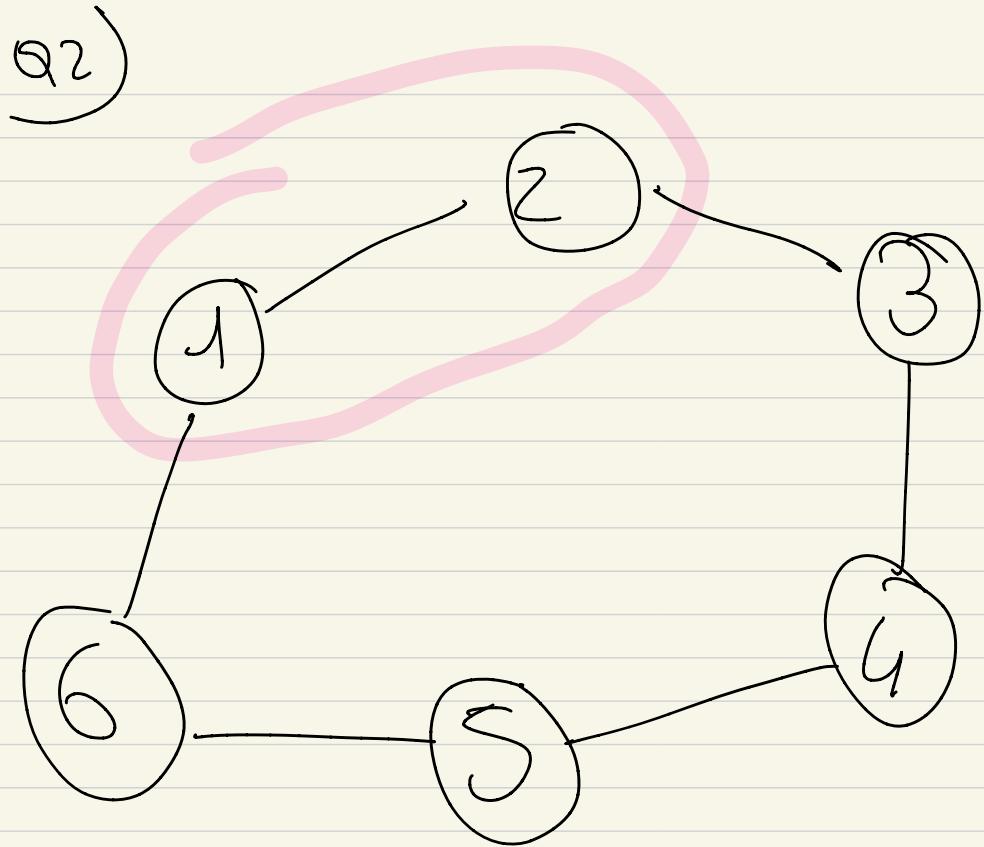
\Rightarrow il y a $|U|$

$$\{x(a), a \in U\} \subset X.$$

$\Rightarrow |U| \leq |X|$

Tips: Dans un raisonnement, ce qui est important c'est de voir où on fait apparaître nos hypothèses.

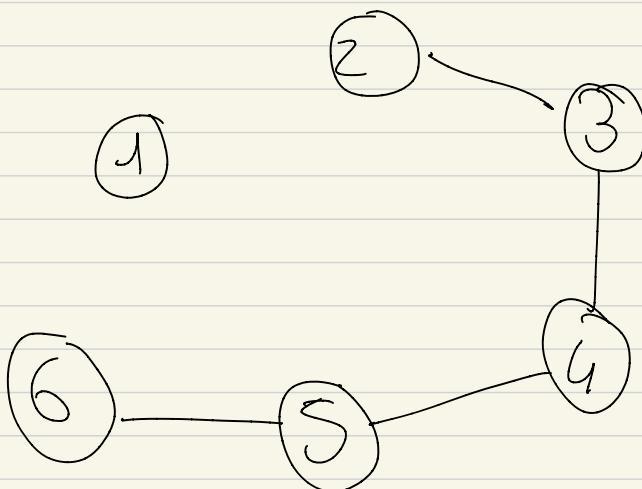
PROUVEDON



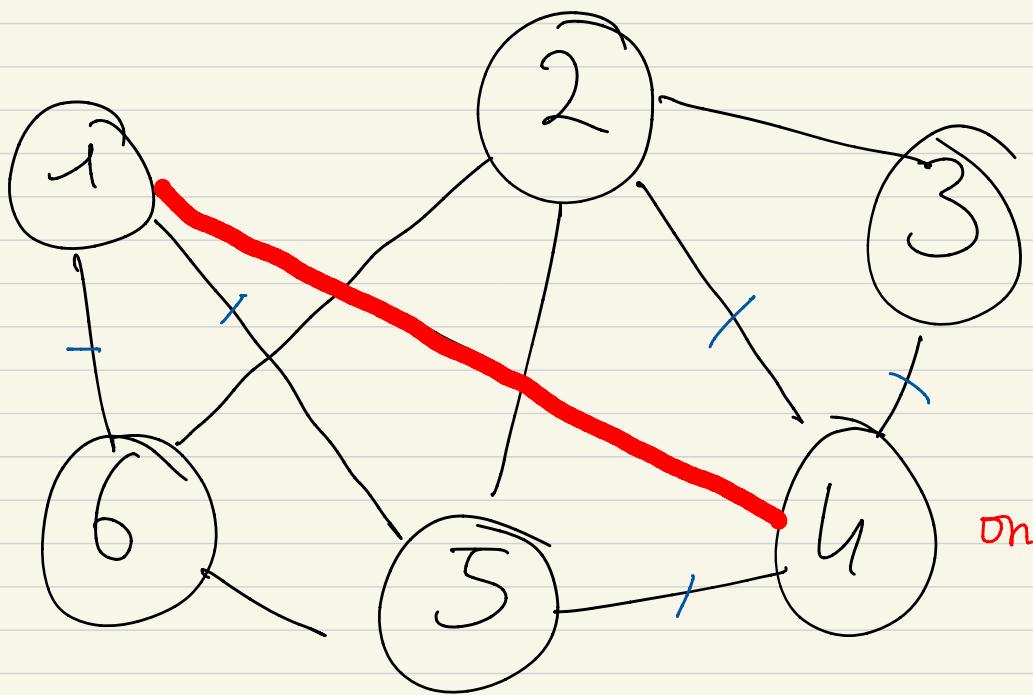
Step 1 : on ajoute 1 à χ

$$\chi = \{1\}$$

$$U = (1, 2)$$



Connexion :



on a commencé
avec $(1,4)$

Héritage, $X = \{1, 4\}$

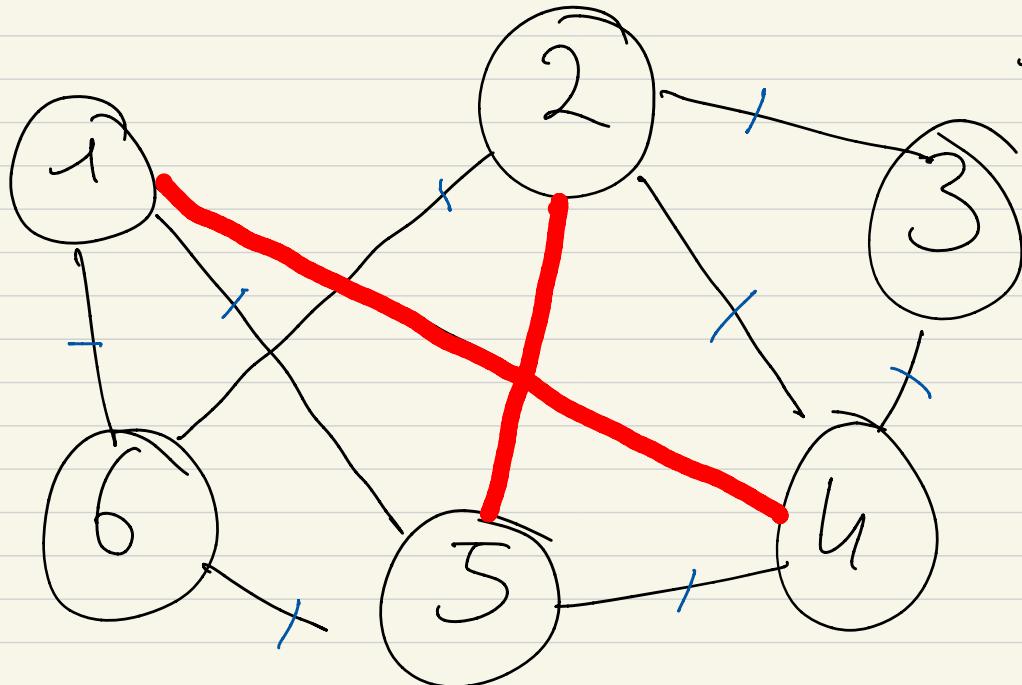
$U = \{(1, 4)\}$

on supprime les arêtes adjacentes

Itérati° 2 : on choisit l'arête $(2, 5)$

$$\rightarrow X = \{1, 4, 2, 5\}$$

$$\rightarrow U = \{\{1, 4\}, \{2, 5\}\}$$



on suppose
le reste des
arêtes adjacentes
à 2 ou 5

Tant que $E' \neq \text{null}$ \rightarrow on choisit une arête

Q3) \rightarrow Si j'prends n'importe quelle arête
du graphe,
+ arête $\{x, y\} \in G$
 x ou y sont dans G

Etape :

Soit $\{x, y\} \in G$

Deux uniques possibilités :

- Soit elle est dans U (car on l'a choisie)
- Soit elle est supprimée

Soit $\{x, y\}$ une arête de G

Celle appartient à E' au début de l'algo

et que E' est vide à la fin de l'algo

\Rightarrow l'algorithme l'a supprimée

- Soit $\{x, y\} \in U$

$\Rightarrow x$ et $y \in X$

- Soit $\{x, y\}$ a été supprimée parce que x ou y était une extrémité d'une arête de $U \Rightarrow$

x ou $y \in X$

\Rightarrow au moins une extrémité de $\{x, y\}$ est dans $X \Rightarrow X$ recouvrement

erreur:

$$|X| = 2 \times |U|$$

\hookrightarrow on va le montrer après, on va montrer que 2 arêtes de $|U|$ n'ont aucune extrémité commune.

Q.U

Q.U : Mg les arêtes de \downarrow sont disjointes

construit

par l'algo

\hookrightarrow j'ai compris l'idée !

chaque fois qu'on choisit une
suite de dans U toutes les
arêtes adjacentes sont éliminées
on ne peut donc ultérieurement choisir
une droite avec une extrémité

comme avec a

→ on pourrait faire aussi un raisonnement
par l'absurde

En exam : il faut sortir les arguments

→ il faut dire où interviennent
les hypothèses.

→ On peut pas dire "c'est vrai dc
c'est vrai"

on peut être
plus concis en
exam !

$$Mg |X| = 2 \times |U|$$

A chaque itération ...

l'algo ajoute les 2 extrémités des arêtes de $|U|$ à X et comme ces arêtes sont disjointes, le nb de sommets de X est 2 fois le nb d'arêtes de $|U|$

Q5) Mises bout à bout, toutes les qsts précédentes vont permettre de conclure.

Prérequis : Savoir ce qu'est un rapport d'approximation

Notre que l'algo a un rapp d'appr. relative égale à 2

→ autrement dit : Mg la solution trouvée (X) est AU PLUS 2 fois pire que la solut optimale (X^*)

nb de sommets de X

càd

$$|X| \leq 2 \times |X^*|$$

$\underbrace{|X|}_{\mathcal{F}(x^a)}$ $\underbrace{|X^*|}_{\mathcal{F}(x^{opt})}$

Correction :

Sat X^* un recouvrement optimal

On veut montrer que

$$\frac{|X|}{|X^*|} \leq 2$$

On sat que $|X| = 2 \leftarrow |U|$

et U ensemble d'arcs disjoints

D'après Q1, $|X^*| \geq |U|$

Donc

$$|X| = 2 \times |U| \leq 2 \times |X^*|$$

$\brace{ \text{on a majoré par } X^* }$

$$\Rightarrow \frac{|X|}{|X^*|} \leq 2$$

$\brace{}$

il contiendra au pire $2x$ trop de sommets

\rightarrow c'est un algo 2 approché

$\brace{ \begin{array}{l} \text{genre } f(X) \\ \text{voir ce que} \\ \text{c'est} \end{array} }$

\rightarrow c'est une garantie de performance

Dans notre exemple, le rapport vaut 1
↳ Donc c'est mieux que la garantie

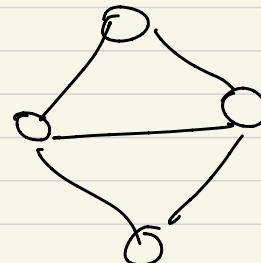
→ essaie de trouver mieux \hat{c} recourent,
tu n'auras \neq mieux

Cas du TSP : on peut pas trouver un
ratio d'approximat^o \neq de l'infini

Nétaheuristiques
à l'examen : qst de cours possibles

COURS :

✓ Étageuristiques



- Descente
- Recuit simulé

- Tabou
- Algos génétiques

Pb Sac à dos :

une solution = un sac d'objets.

1 sac = vecteur binaire de n bits
 ↗ objet 1 dans le sac

[1 1 0 1 1 0]

Un voisin = un vecteur qui ressemble

↳ ex de solut° proche :

→ solut° avec 1 bit inversé

Voisins = Solutions réalisables avec 1 bit de différence

[0 1 0 1 1 0]

PENSER l'ens. des solutions comme UN RÉSEAU de SOLUTIONS

AG :

→ un peu ≠ du reste

→ Algo Descente + Monte Carlo

" → À partir d'une solut° initiale,
regarder si un voisin proche
n'améliore pas la fct objectif."

① choisir une solution initiale

→ l'algo ne dit pas
comment la choisir

② aller de voisin en voisin

tant qu'on améliore la fct objectif
ex : pour le sac à dos (pb de MAX)

→ on aura tendance à minimiser
(descendre).



ici que j'aille à gauche ou à droite, mes voisins
sont + grands, on est bloqués ds 1 bphi

Local.

→ Alors que l'opti global est + loin

Solution: "on fait ça plein de fois
sur des solut° initiales
tirées aléatoires puis ..."

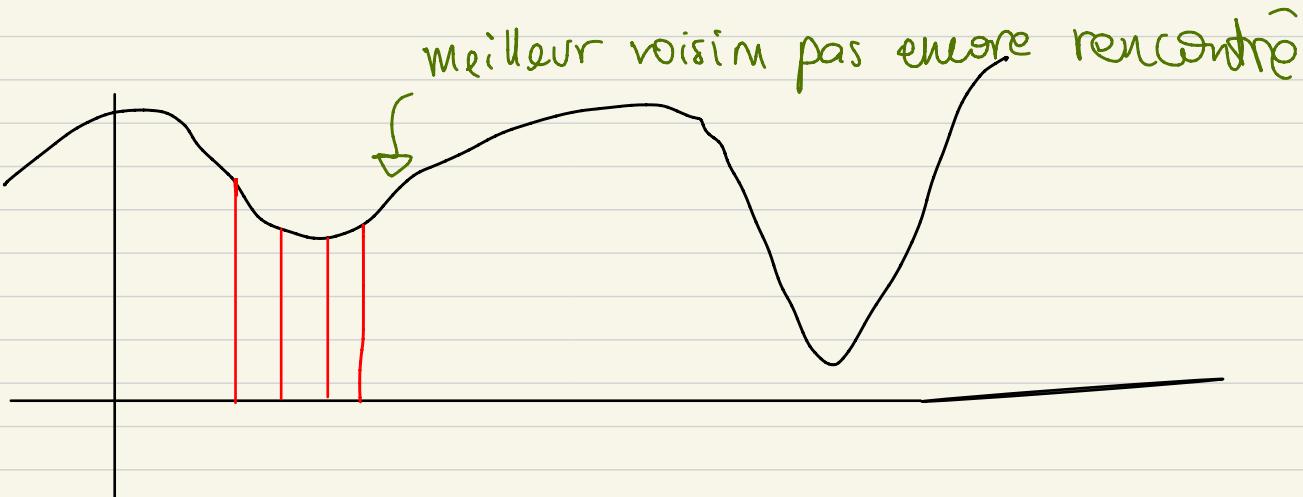
Méthode de Monte Carlo

- ① Tirer un grand nombre de solutions initiales
- ② Pour chacune, faire une méthode de descente
- ③ Garder la meilleure solut° obtenue

Autre idée : s'autoriser à renoncer de temps en temps

→ Tabou et Recuit simulé

① Tabou



Dans le pb du sac à dos, on a n voisins, pas seulement 2.

① s'autoriser à renoncer vers un voisin à condition que je ne l'ai pas déjà rencontré.

→ implique : temps de calcul considérable

pour explorer l'ens. des solut°, ainsi
que de la mémoire pour mémoriser
les solutions déjà rencontrées

→ donc la liste Tabou* maintenue par la
méthode tabou est limitée

(* on a pas le droit d'y aller.
↳ risque : rebouclage

Ces solutions :

Descente, Nc, Tabou :

- y'a pas vraiment de limite de temps de calcul
- mais ça peut donner des très bonnes solutions
en un temps maîtrisé par le programmeur.

Recuit Simulé

→ vient du monde de la physique

- Avez vous déjà fait du caramel ?

→ Nette feu doux

→ Sucre fond

→ On le sort du feu

→ On le met sur une assiette

→ Le sucre dure en un caramel

↳ "il cristallise"

Analogie : le sucre change en fonction de la température

Phase 1 : C'est les particules de sucre, on a le droit de bouger beaucoup, tant que la température est forte

Phase 2 → on mesure que la température baisse,
on bouge ☺

Méthode Reuit simulé (boucle)

problème de minimisation

Paramètre T (température)

$$\begin{aligned} T &= T_0 \\ x &= x_0 \text{ solution initiale} \end{aligned}$$

Repter

Repter N_r itérations

Choisir au hasard un voisin x' de x

si $f(x') < f(x)$ c'est une ^{solution} meilleure que x

dans le réseau
de solut°
avec un
réseau
de voisins

sinon $x = x'$ avec une probabilité p_T

$$T = T - \text{pas}$$

p_T décroît quand
 T diminue

⇒ de façon surprenante : cet algo MARCHE !

Cela s'appuie sur: ∃ un moyen de faire descendre la température

lentement et que les itérations sont suffisantes
→ ce processus converge vers une solut° optimale

Autres paradigmes

- Algo du Kangourou (méthode de descente et saute !)
 - Algo Fourmi (manière dont les fourmis vont chercher de la nourriture
 - phéromones laissées sur le chemin de la bouffe)
- |
 → cet algo a été transposé en opti-combinat.
- tirage des solut° initiales : fourmis
 - garder en mémoire : phéromones !

En pratique → ça marche très très bien dans l'industrie

Méthodes du voisinage

Condition de faisabilité : réseau des solutions est connexe

pour pouvoir aller
de n'importe
quelle solut° à
une autre
en plusieurs
mouvements successifs

Pb sac à dos :

(1 00 11 00)

Voisins \rightarrow vecteurs binaires avec 1 bit
de différence ; solut° FAISABLE

Ex :

	1	2	3	4	5	6	7
R _i	3	4	2	1	3	1	4
w _i	8	10	20	10	20	30	10

$P = 9$

En de voisins. (on a n voisins au max)

↪ voisins par enlevage

(0001100) (1000100) (1001000)

Pas voisin :

(1101100)

voisins

↪ par ajout :

(1011100)

(10011110)

→ Pq ce system de voisinage est-il connexe ?

↪ Faire une preuve. $\Delta \rightarrow$ preuve constructive.

→ on doit prendre 2 sommets et montrer qu'on peut construire un chemin.

↪ décrire comment il est fiché.

→ tous les sommets intermédiaires doivent être faisables !

Applicat° de l'algo:

$$x = (1001100)$$

Vous → rectas binaires avec 1 bit de différance faisable.

$$y = (1010001)$$

si y est faisable toute solution avec les mêmes 0 et quelques 0 en plus est faisable à la place de 1 faisable

1	2	3	4	5	6	7
3	4	2	1	3	1	4
8	10	20	10	20	30	10

$$P = 9$$

Partir de x
élever les 1, un par un
de moins en moins
jusqu'à obtenir des zéros partiel
et ensuite ajouter un à
un les 1 correspondant
au sac y .

$$(1001100)$$

$$(0001100)$$

$$(0000100)$$

$$(0000000)$$

$$(1000000)$$

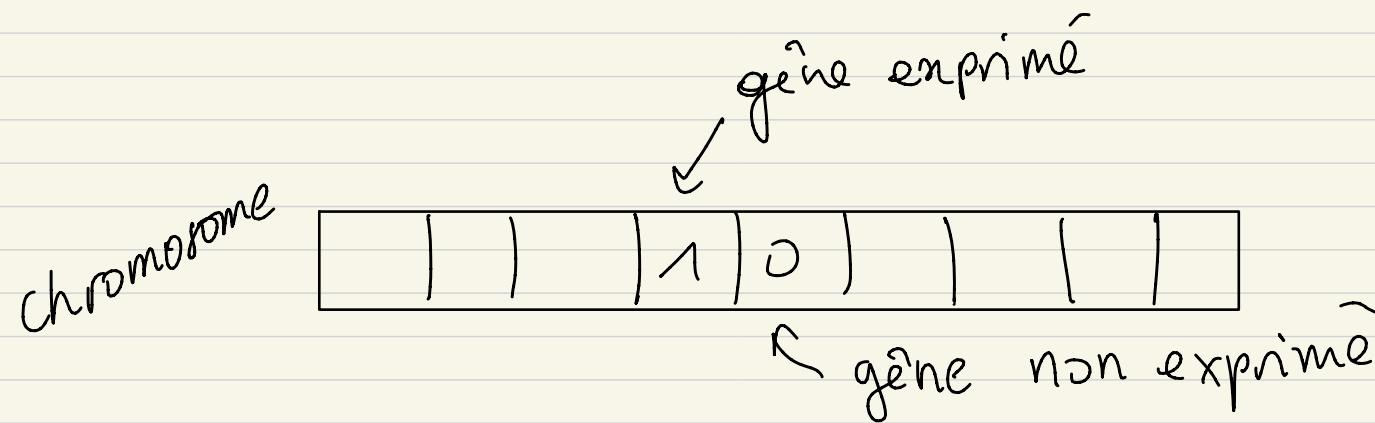
$$(1010000)$$

$$(1010001)$$

cet algo c'est Ø de la friche car on veut un algo qui marche à tous les coups

Algorithme génétique :

→ inspiré de Darwin : population qui va évoluer :



Opérations :

Mutation : aléatoire

changer

Croisement :

PAPA

A	A'
---	----

NANAN

B	B'
---	----

CROSS-OVER

A	B'
---	----

B	A'
---	----

en : $x = (1 \ 00 \mid 11 \ 00) \quad y = (1 \ 01 \mid 0001)$

But
final
grouped
la
bonne
solution

CROSS OVER :

$$(\overbrace{100}^A \quad \overbrace{0001}^{B'})$$

$$(\overbrace{101}^B \quad \overbrace{1100}^{A'})$$

Sélection : quels individus éliminer ?